REPORT DOCUMENTATION PAGE

2 Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY) 26-03-2014	2. REPORT TYPE Ph.D. Dissertation	3. DATES COVERED (From - To) -

4. TITLE AND SUBTITLE Model-based compositional design of networked control systems	5a. CONTRACT NUMBER W911NF-10-1-0005
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 106011
6. AUTHORS Eyisi, Emeka P.	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES Vanderbilt University PMB # 407749 2301 Vanderbilt Place Nashville, TN 37240 -7749	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSOR/MONITOR'S ACRONYM(S) ARO 11. SPONSOR/MONITOR'S REPORT NUMBER(S) 56919-NS-DPS.40

12. DISTRIBUTION AVAILIBILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

The rapid advancement in digital technology over the past few decades has fueled the progress in computation and communication technologies. Enabled by this progress, complex engineered systems commonly referred to as Cyber-Physical Systems (CPS), resulting from the integration of computing, communications and control, and in direct interaction with the physical world, are becoming ubiquitous in our daily lives. Examples of these systems include process control, automotive systems, networked robotics, medical systems, electrical power grids and

15. SUBJECT TERMS

Passivity-based compositional design, Networked control systems

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Yuan Xue
a. REPORT UU	b. ABSTRACT UU	c. THIS PAGE UU	UU		19b. TELEPHONE NUMBER 615-322-2926

Report Title

Model-based compositional design of networked control systems

ABSTRACT

The rapid advancement in digital technology over the past few decades has fueled the progress in computation and communication technologies. Enabled by this progress, complex engineered systems commonly referred to as Cyber-Physical Systems (CPS), resulting from the integration of computing, communications and control, and in direct interaction with the physical world, are becoming ubiquitous in our daily lives. Examples of these systems include process control, automotive systems, networked robotics, medical systems, electrical power grids and environmental monitoring systems among others. These real world CPS are increasingly being monitored and controlled by networked control systems (NCS) and are often employed in critical settings, therefore the assurance of properties such as stability, performance, safety and security are essential. As a result, the analysis and design of NCS architectures have recently gained increasing attention. This dissertation addresses several fundamental challenges in the modeling, design, analysis and evaluation of dependable networked control systems.

First, a domain specific modeling language (DSML), Passive Networked Control Systems (PaNeCS), is presented. PaNeCS raises the level of abstraction of NCS design and allows automated analysis, code generation, system configuration, deployment, and testing. PaNeCS is based on passivity and ensures the "correct-by-construction" design of NCS by enforcing passivity constraints on the components of the NCS as well as their interconnections. Simulation and experimental models generated from the tool are presented to demonstrate the robustness of NCS designs using the tool. Second, an integrated passivity-based adaptive sampling control (PBASC) architecture is presented. PBASC architecture addresses the challenges due to the limited network resources as well as the presence of network uncertainties. The underlying idea of PBASC architecture is to simultaneously allow the variability of sampling intervals as well as ensure stability. Hence, in the proposed framework, while passivity ensures the robustness of the NCS in the presence of uncertainties, adaptive sampling ensures the efficient utilization of network resources. Third, an integrated modeling and simulation tool, Networked Control Systems WindTunnel (NCSWT), based on High Level Architecture (HLA), is introduced. NCSWT integrates Matlab/Simulink and ns-2 for the accurate and efficient evaluation of NCS. Finally, an energy-based attack detection (E-BAD) approach for network control systems is presented. E-BAD is a contribution towards ensuring security of NCS. The underlying approach is based on using the fundamental notion of a system's energy balance in the detection of malicious attacks in NCS. The impact of various attack models on NCS are characterized providing conditions for passive as well as non-passive attacks. Simulation and experimental results are presented in order to evaluate the proposed detection mechanism.

MODEL-BASED COMPOSITIONAL DESIGN OF NETWORKED CONTROL SYSTEMS

By

Emeka P. Eyisi

Dissertation
Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
of the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

December, 2013

Nashville, Tennessee

Approved:
Professor Xenofon Koutsoukos
Dr. Shige Wang
Professor Gabor Karsai
Professor Yuan Xue
Professor Janos Sztipanovits

To my amazing wife, Nkiru and my sons Kosi and Kesi,

your abundant support, love and laughter kept me going through this entire journey.

"Onye kwe Chi ya ekwe, Ekene diri Chukwu."

ACKNOWLEDGMENTS

During my graduate school at Vanderbilt, I had the great pleasure of working with an amazing group of talented people. Most especially, I would like to thank my adviser Professor Xenofon Koutsoukos for his guidance and advise with this work, and for always trying to put the research problem into perspective. I am grateful to Professor Yuan Xue, Professor Janos Sztipanovits and Professor Gabor Karsai, who have always kept an open door policy, giving me the opportunity to stop by at any time for a discussion. I would like to thank Dr. Shige Wang, who I can always rely on for advice both on research related matters and personal life in general.

I will like to thank my fellow colleagues who have affected every aspect of my study at Vandy both work related and outside work activities, in particular Heath LeBlanc, Zhenkai Zhang, Jia Bai, Joe Porter, Ashraf Tantawy, Daniel L. C. Mack, Stephanie Weeden-Wright, Siyuan Dai, Chetan Kulkarni, Peter Horvath, Mark Yampolskiy, Gabor Simko and members of the ModEs and VaNets group.

I would also like to thank my parents, siblings and in-laws for their continuous encouragement.

Most importantly, I would love to thank my wife Nkiru, for her endless support, love and guidance. And to my sons, Kosisochi and Kesiyonna, thanks you for always putting a smile on my face no matter how bad my day might be going. Finally, this work would not have been possible without the financial support of the National Science Foundation (CNS-1238959, CNS-1035655, and CCF-0820088), US Army Research Office (ARO W911NF-10-1-0005), and Lockheed Martin.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

The rapid advancement in digital technology over the past few decades has fueled the progress in computation and communication technologies. Enabled by this progress, complex engineered systems commonly referred to as ***Cyber-Physical Systems*** **(CPS)**, resulting from the integration of computing, communication and control, and in direct interaction with the physical world, are becoming very ubiquitous in our daily lives [1]. Examples of these systems include process control, automotive systems, networked robotics, medical systems, electrical power grids and environmental monitoring systems etc. These real-world CPS are increasingly being monitored and controlled by ***Networked Control Systems*** **(NCS)** and are often employed in critical settings, therefore the assurance of properties such as stability, performance, safety and security are essential. As a result, the design of NCS architectures has recently gained increasing attention and has become an area of active research.

NCS can be described as control architectures which consist of plants (the systems or processes to be controlled), sensors (components that collect information on the controlled plant states), controllers (computing units where the control algorithms run) and actuators (components that apply the computed control signals), whose operations are spatially distributed and coordinated through information exchange over a communication network [2]. Although, NCS have existed for several decades, the advancement in technology has resulted in the paradigm shift from the classical point-to-point architecture, that is a wire connects a central computer with each sensor or actuator point, to more modular architectures comprising multiple nodes communicating with each other over communication networks, which are more suitable for real-world CPS.

The current paradigm shift in NCS architectures presents numerous benefits such as improved efficiency, flexibility in system design and reliability through significantly reduced installation, reconfiguration and maintenance time and costs as well as ease of future expansion [3]. These benefits have resulted in even further expanded roles for NCS applications.

1.2 Research Challenges

While NCS certainly provide numerous benefits, the complexity of their modeling, analysis and design raises several challenges which need to be overcome to fully utilize the benefits. The most significant challenge in the design of NCS is the ***impact of network and platform effects***. The network effects consist of network uncertainties, such as time-delays and the possible loss of packets, introduced due to the use of a communication network while the platform effects are mainly as a result of the implementation of the control algorithms which may be subject to quantization. These network and platform effects pose considerable concerns due to their significant impact on overall system stability, performance and safety.

A second challenge in NCS is due to the ***limited network resources*** and therefore limited information exchange over the communication network. Increasing the availability of the plant's current information to the controller often leads to a better performance of the overall control system but due to the constrained network resources this is not always possible. This limited availability of sensor and control data packets has a direct impact on the achievable stability and performance of the NCS. How to attain the desirable trade-off between satisfying the communication constraints as a result of the limited network resources without sacrificing stability and ultimately performance of the overall system is a challenging problem.

Due to the complexity of NCS, the ***modeling, analysis and evaluation*** of system properties and behaviors under various conditions present another significant challenge. When building real-world systems, typically, models of the systems are first created and extensively analyzed and evaluated using simulations and experiments on various prototypes before the deployment of the actual systems. The complexity of NCS makes it difficult to determine the correct level of abstraction at which to model these systems in order to enable effective and efficient analysis and evaluation to ensure they satisfy the design requirements.

Finally, the introduction of the communication network and the often distributed nature of NCS make these systems vulnerable to malicious attacks and pose security concerns. NCS expose multiple entry points through which adversaries can exploit potential vulnerabilities and introduce malicious artifacts. The presence of such attacks can be potentially catastrophic considering the safety-critical nature of most NCS. ***Designing dependable and secure*** NCS that can guarantee safety or a specified level of performance is a very important and challenging task. Understanding the areas of vulnerabilities in NCS as well as determining effective designs and strategies to detect the presence of attacks and techniques to handle these attacks is daunting, considering the complexity of NCS.

1.3 Overview and Contributions

In this section, we provide an overview of the dissertation and the contributions towards addressing the outlined fundamental challenges involving the use of NCS architectures in the construction of CPS. The technical contributions are contained in Chapters 4-7.

Chapter 2

This chapter presents the related work in networked control systems. We provide a background on NCS architectures and passivity-based techniques. We review the various adaptive sampling strategies. We also discuss the various approaches for the evaluation of NCS using simulation and the integration of simulators. Subsequently, we discuss the dependability of NCS and provide a background on model-based detection. In addition, we review the recent work towards securing NCS. Due to the importance of experimental platforms in evaluating NCS designs, we present a few NCS applications related to our work. Finally, we provide a summary describing the relationship of the related work to the research presented in this manuscript.

Chapter 3

A background on the concepts of dissipativity and passivity is presented in this chapter. We discuss the properties of passive systems as well as the various passivity preserving interconnection rules. We introduce the concept of passivity indices for quantifying the degree of a system's passivity and the use of passivity indices in rendering non-passive systems passive. Additionally, we discuss the wave variable transformation, a concept closely related to passivity.

Chapter 4

The tight integration of the design layers in CPS makes it extremely difficult to guarantee global system properties such as stability or a defined notion of performance. Additionally, the heterogeneity makes it extremely challenging to apply model-based techniques through out the design process. In order to address these challenges, in this chapter we introduce PaNeCS, an end-to-end tool-chain for the analysis and design of passivity-based NCS that are robust to network effects. The underlying idea of PaNeCS is that by imposing passivity constraints on the component dynamics, we can then apply model-based techniques to simplify the design of NCS with stability guarantees, even in the presence of network uncertainties. The specific contributions in this chapter are the following:

- In [4, 5, 6], using Model Integrated Computing(MIC) together with the concept of passivity, we developed PaNeCS, a domain specific modeling language (DSML) for the compositional modeling and design of passivity-based networked control systems. PaNeCS is integrated with a component-based analysis tool for checking passivity of NCS components (linear systems) using linear matrix inequalities (LMIs). Using the object constraint language (OCL) [7], PaNeCS enforces a set of structural compositional rules in the modeling of NCS. The combination of the compositional rules and the component-based analysis tool ensures a "correct-by-construction" of passivity-based NCS designed in PaNeCS. The resulting designs are by construction robust to implementation effects and network uncertainties.

- We developed a model interpreter that generates platform-specific simulation code from a PaNeCS model, for NCS simulation in Matlab/Simulink and TrueTime in order to evaluate the designed NCS under various network conditions. A case study on the control of multiple linear plants over a wireless network is presented in order to evaluate the tool.

- We have also developed a model interpreter for the generation of executable from a PaNeCS model, for running experiments involving a class of nonlinear Euler-Lagrangian systems (such as networked multi-robot systems). We presented an experimental case study involving the control of a networked multi-robot system.

Chapter 5

In this chapter, we present a passivity-based adaptive sampling control (PBASC) architecture, our contribution towards addressing the challenges due to the limited network resources. The underlying idea of PBASC architecture is to simultaneously allow the variability of sampling intervals as well as ensure stability. The specific contributions of this chapter are highlighted as follows:

- We developed an approach that integrates two control theoretic concepts, passivity and adaptive sampling, and applied the integration to a trajectory tracking problem in a hierarchical NCS [8]. In the integration, the passivity-based approach guarantees stability of the NCS in the presence of network uncertainties while the adaptive sampling not only ensures the efficient utilization of network resources but also provides the flexibility of incorporating network scheduling adaptation. The integrated framework was used in [9, 10], for a joint design of sampling rate adaptation and network scheduling in distributed networked control systems.
- We developed a pair of sample-and-hold components, a variable passive sampler and a variable passive hold, that facilitates the integration of sampling-and-hold mechanism with variable sampling intervals while maintaining passivity.
- We integrated an example sampling policy based on self-triggered control using a performance criteria defined as a function of the error, in order to compute sampling intervals.
- Additionally, we demonstrated our approach with a case study on the trajectory tracking control of a robotic manipulator over a wireless network. We provide simulation results using Matlab/Simulink/TrueTime. We also provide experimental results using an actual robotic manipulator and a wireless network. We compare our approach to the case of a fixed sampling period. We show that the robotic manipulator tracks a desired trajectory while reducing the utilization of network resources compared to the case of a fixed sampling period. We also provide results to demonstrate the robustness of our approach under various network uncertainties such as time-varying delays and packet loss.

Chapter 6

This chapter presents an integrated simulation framework for the effective and efficient evaluation of network control systems. The accurate evaluation of NCS is challenging considering the heterogeneity of the design layers of NCS and the potentially different timing semantics of the simulators. In order to address these challenges, we introduce NCSWT, a tool-chain for the modeling and simulation of networked control systems based on the High Level Architecture (HLA), which guarantees the accurate time synchronization and data distribution of the integrated simulation. Using MIC techniques, NCSWT provides efficient modeling and integration of NCS components as well as the generation of software components used in the simulation of the designed NCS. The specific contributions in this chapter are highlighted as follows:

- We developed NCSWT[11, 12, 13], an integrated modeling and simulation tool which combines two defacto simulation tools in their respective domains for the simulation of NCS. The integrated simulators are Matlab/Simulink for the modeling, design and simulation of control systems, and ns-2 for modeling and simulation of the communication networks.
- We developed three DSMLs. NCSWT MIL for the definition of components and information exchange of NCS in terms of HLA-based concepts. CDML is designed for the integration of control design concepts of the NCS while NDML is designed for the integration of the network components of the NCS.
- We developed model interpreters integrated in the three DSML for the generation of software components for the simulation of NCS.
- We provide a run-time environment for the simulation of NCS using Matlab/Simulink for the control design component and ns-2 for the network components of the NCS.
- We present case studies for the modeling and simulation of NCS under various realistic network conditions such as packet losses and time delays in order to illustrate the tool. NCSWT has also be used to evaluate the NCS approaches developed in [9, 10, 14].

Chapter 7

In the previous chapters, the possibility of malicious attacks on NCS architectures was neglected. In this chapter, we explicitly consider the possibility of cyber attacks on NCS infrastructures. We introduce an energy-based approach for the detection of malicious attacks on NCS. The contributions presented in this chapter are summarized as follows:

- We formulate the detection of attacks using the system property of energy. We define the notion of energy-based monitor for networked control systems and introduce an energy-based attack detector (EBAD), a monitor based on the intuitive concept of a system's energy.

- We prove that EBAD solves the attack detection problem and also illustrate that the impact of attacks on a system can be estimated as the excess or loss in a system's energy. Based on estimated energy, we qualitatively characterize the attack as being either passive or non-passive.
- Based on well-known attack models, we illustrate the impact of the defined attacks on the energy of a system using the energy-based formulation. We also present analytical results to show conditions in which the attacks can violate passivity properties of the overall system.
- We evaluate the proposed methodology using simulations as well as experiments on a robotic manipulator test-bed focusing on the velocity tracking ability of a single joint over a network. The results show that the detector is effective in detecting the presence of attacks on NCS in addition to characterizing the impact of the attacks.

Chapter 8

This chapter provides a summary of the work contained in the manuscript, and describes the future directions in which the current research could be improved.

CHAPTER 2

RELATED WORK

NCS is a multi-disciplinary research area involving concepts and methods from control, communication, operations research, computer science and management science. In this chapter we discuss works that are relevant and directly related to the contributions of this dissertation. In particular, we focus on passivity-based design of networked control systems, adaptive sampling, model-based design approaches, NCS simulation frameworks, dependability and representative NCS applications. We provide an introduction of the main configurations and categories of NCS as well as a review of teleoperation and passivity-based NCS in Section 2.1. We introduce the concept of adaptive sampling in Section 2.2. Section 2.3 discusses model-based design techniques and provides a review of the various approaches for simulating NCS. Section 2.4 introduces the concept of dependability and model-based detection, in addition to a review of security in NCS. We describe representative NCS applications in Section 2.5. Finally, Section 2.6 discusses how the contributions of the dissertation compares to the scope of the literature.

2.1 Networked Control Systems

In this section, we first introduce the main configurations of NCS and then we describe the categories of NCS. Focusing mainly on the *control over networks* category of NCS, we describe the various techniques for addressing the various network effects. We then provide a review of existing works in passivity-based NCS.

2.1.1 Configurations of Networked Control Systems

The two general configurations in NCS are the *direct structure* and *hierarchical structure* [15, 16, 17]. These configurations are described as follows:

2.1.1.1 Direct structure

The *direct structure* is the more common of the two structures. In this configuration, the NCS is composed of a controller and a remote system containing a physical plant, sensors and actuators as depicted in Figure 1. The controller and plant are typically located at different locations and are linked by a data network in order to perform closed-loop control. This structure is widely used in several application such as DC motor speed control [18] and distance learning lab [17].

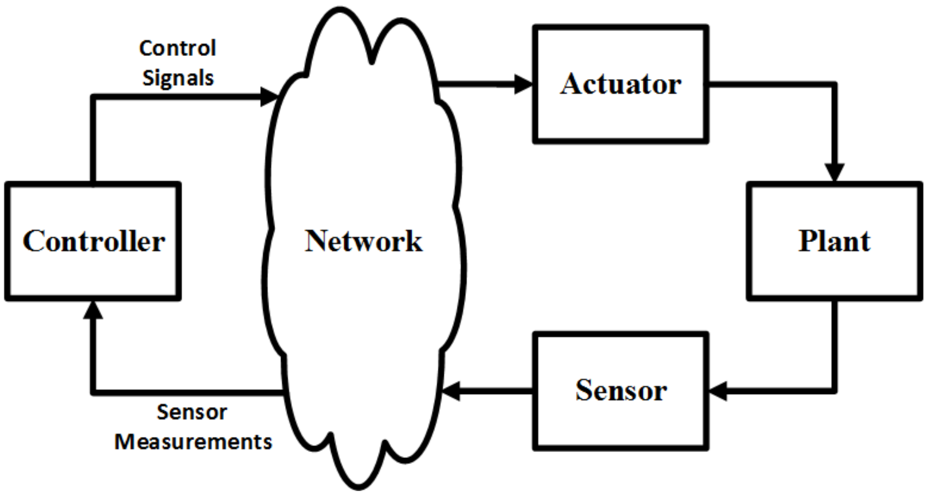


Figure 1: Direct Structure of NCS

2.1.1.2 Hierarchical structure

The *hierarchical structure* typically consists of a main controller and a remote closed-loop system as depicted in Figure 2. The main difference between the direct and hierarchical structure is the presence of a local controller in the hierarchical configuration. In a typical operation in this configuration, the main controller computes and sends control information via a network to the remote system. The local controller of the remote system then processes the information to perform local closed-loop control. This structure is widely used in several applications such as robotic manipulator control [19, 20], mobile robots [21], teleoperation [22, 23] and unmanned aerial vehicles (UAVs) [24, 25] etc.

The use of either structure is based on the application requirements as well as the preference of the designer. Clearly, the control and analysis of the direct structure can be applied to that of the hierarchical structure by simply treating the remote closed loop system purely as a plant.

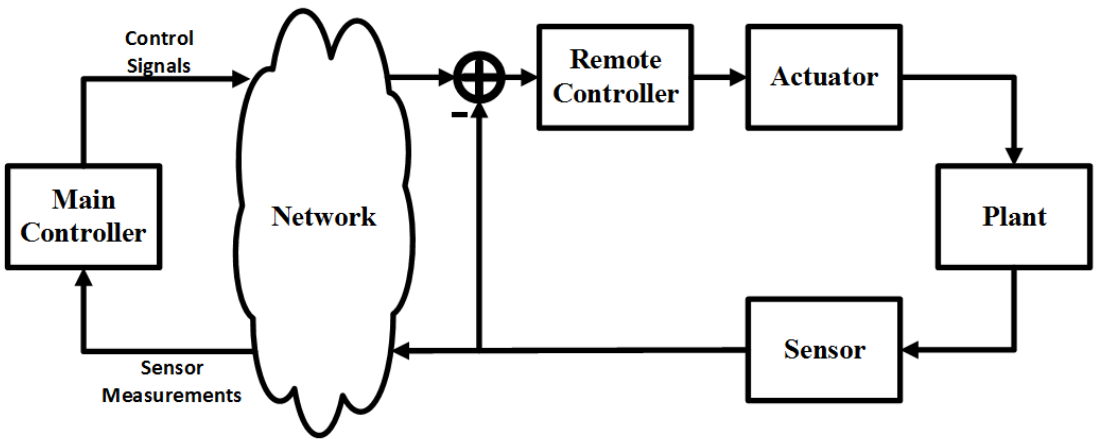


Figure 2: Hierarchical Structure of NCS

2.1.2 Classification of Networked Control Systems

Research in the field of NCS can be generally categorized into three main areas namely *multi-agent systems*, *control of networks*, and *control over networks* [26]. We will describe each of the categories but we will focus mainly on the control over networks.

2.1.2.1 Multi-agent systems

Multi-agent systems deal with how network architectures and interactions between networked components influence global control objectives. The main problem associated with multi-agent systems is mainly to understand how local laws describing the behavior of the individual agents impact the global behavior of the overall networked system. Research in multi-agent systems can be further categorized into two main areas of active research.

1. The first area involves the design and analysis of distributed estimation techniques. The application of this area is the framework of sensor network technologies. There's a growing need for distributed collection and processing of information measurements using tools and algorithms that provide high performance in terms of online estimation. The requirements for these algorithms are often to reduce communication load among sensor nodes and robustness to packet losses and node failures.

   The distributed estimation for sensor networks has been a very active area of research in the field of communication theory, computer science and signal processing [27, 28], it is only in the past few years that considerable attention has been devoted to this area within the control community. The authors in [29] first related the the consensus problem in a distributed setting to the distributed estimation problem. Their idea resulted in a new approach to distributed Kalman filtering as described in [30, 31]. The work in [32] provides an interesting survey on this subject. Various extensions such as to switching topologies [33, 34] and to randomly switching topologies [35] have also been developed. The work in [36] provides a nice discussion in regards to the use of sensor networks in control applications. Additionally, other works in this area involve designing resilient distributed multi-agent systems in the presence of adversaries as described in [37, 38] and references therein.

2. The second area of active research in multi-agent systems involves the control of autonomous agents such as robots and unmanned vehicles. Groups of autonomous agents with computing, communication, and mobility capabilities are becoming economically feasible and can perform a variety of spatially distributed sensing tasks, such as search and rescue, surveillance, environmental monitoring, and exploration. In typical examples of motion coordination problems, groups of autonomous agents require the ability to cover a region of interest, to assume a specified formation, to rendezvous at a common point, or to move in a synchronized manner as in flocking behaviors.

   The literature on the control of group of autonomous agents is exorbitant and rapidly growing. An important contribution towards a network model for mobile interacting agents was introduced in [39] for the case of mobile robots. This model consists of a group of identical distributed anonymous mobile robots whereby no explicit communication takes place between them, and at each time instant of an activation schedule, each robot senses the relative position of all other robots and moves according to a pre-specified algorithm. In [40], the authors analyzed the communication complexity for control and communication algorithms in multi-agent network of robots. [41, 42] provided a comprehensive set of models and the time complexity analysis of numerous algorithms. Over the years, there has been a lot of progress in regards to distributed motion coordination algorithms, such as in pattern formation [43], flocking [44, 45], swarm aggregation [46, 47], self-assembly [48], deployment [49], rendezvous [50] and cyclic pursuit [51].

2.1.2.2 Control of Networks

Control of networks is mainly concerned with providing a certain level of performance or *quality of service* (QoS) to a network data flow, while achieving efficient and fair utilization of network resources. The control of networks is a very active research area. The ability to measure and modify network parameters is typically a fundamental requirement in order to be able to control network performance.

The fundamental problems of interest in the area of control of networks are scheduling, call admission, routing, power control, flow control and various other resource allocation problems. There has been some significant progress in the theoretical understanding of network congestion control for example in the the seminal work by [52]. By explicitly modeling the congestion measure signal fed back to the sources, and by posing the network flow control as an optimization problem where the objective is to maximize the total source utility, it is shown that the rate control problem can be solved in a completely decentralized manner [52, 53]. It is often important to assess the dynamical properties, such as stability and convergence, of the schemes in order to ensure the system will reach and maintain a favorable equilibrium. In most of the existing works, the effect of network delay is ignored. The assumption is typically that the price information, which reflects the cost of network resource usage, is available instantaneously at the source, whereby the sources take immediate action, and that the new rates affect the link prices instantaneously. However, stability of the protocols in equilibrium depends critically on the feedback delay and on the interactions between the protocol implementations and the network dynamics.

Some of the recent works in the area of congestion control include development of mathematical models for flow control under various Internet protocols [54, 55, 56]. Scalable and distributed optimization algorithms have been developed for these control systems [57, 58]. The impact of nonlinearities and time delays in the network flow models have also been considered in [59, 60, 61, 62]. From a resource control perspective, wireless networks are often considered, this is due to the fact that whereas the link capacities in wireline networks are fixed, the capacities of wireless links can be adjusted by the allocation of communication resources, such as transmit powers, bandwidths, or time slot fractions, to different links. The adjustment of resource allocation changes the link capacities, influences the optimal routing of data flows, and alters the total utility of the network.

Hence, optimal network operation can only be achieved by coordinating the operation across the networking stack. This is generally referred to as *cross-layer design*. Resource allocation and congestion control are becoming increasingly important [63, 64, 65, 66];

2.1.2.3 Control over Networks

Control over networks deals with the design and analysis of feedback methodologies adapted to control systems in which control data is exchanged over unreliable communication links. The main objective of control over networks is to provide *quality-of-control* (QoC), which is the performance delivered by the closed-loop operation distributed over a communication network. As described in Section 1.2, although there are several advantages in using NCS, there exist some drawbacks due to the presence of the communication network. Numerous techniques aim to formally characterize NCS properties such as as stability and performance [67, 68, 69]. Research in this area aims to achieve a specified QoC while at the same time handling the network effects including network-induced delays and data dropouts.

Several works have investigated design methodologies for addressing network delays in NCS. In [70, 71], the authors proposed an augmented deterministic discrete-time model methodology which essentially augments the state space of a control system in order to handle network delays, the methodology was extended to handle non-identical sampling periods of a sensor and a controller in [72]. The works in [73] developed a deterministic predictor-based approach to control NCS in the presence of delays. The idea is to reshape the random network delays such that the NCS becomes time-invariant. The approach then uses observers in order to estimate the plant state and a predictor to compute the predictive control based on past output measurements. Nilsson in [74] proposed an optimal stochastic methodology to control NCS on random delay networks. This approach treats the effects of random network delays in NCS as a Linear-Quadratic-Gaussian (LQG) problem which typically aims to minimize a specified cost function.

In other approaches, [75, 76] used non-linear and perturbation theory to formulate network delay effects in NCS as the vanishing perturbation of a continuous-time system under the assumption that there is no observation noise. This approach is restricted to priority-based networks and systems with small sampling periods. In [77], the authors proposed a sampling time scheduling methodology which appropriately selects a sampling period for NCS such that network delays do not significantly affect the control system performance and the NCS remain stable. The authors in [78] proposed a networked controller design in the frequency domain using robust control theory. The major advantage of this approach is the lack of the need for a priori information about the the probability distribution of network delays. In [79], the authors proposed a fuzzy logic modulation methodology for NCS with a linear plant and a modulated PI controller to compensate the network delay effects based on fuzzy logic [80]. The PI controller gains using this methodology are externally updated at the controller output with respect to the system output error caused by network delays. In [69], a survey on the state of the art design methodologies that take into account the effects of packet losses is presented. In [68], packet loss between the controller and the actuator is considered and the separation principle is applied but only for TCP-like communication protocols where packet loss acknowledgment is available unlike in the UDP-like protocols.

The research efforts in [81, 82] have investigated the effects of quantization at length focusing mainly on control and stabilization. In [81], the authors presented a technique to address delays and quantization using a unified framework. The work in [83] proposed a method which aims at decoupling the control design from the implementation layers. This methodology allows the design of state-feedback controllers that minimize a quadratic performance bound for a given level of delay using linear matrix inequalities and also allows the truncating the coefficients of the controller while guaranteeing that a given set of performance constraints is met. The work in [84] adopted this approach in studying the performance degradation caused by time-varying delays.

In addition, other control approaches have been developed to tolerate and compensate for the impact of various network uncertainties in NCS. Some of these approaches include the development of new control paradigms to improve performance and security of NCS whereby the network itself acts as a controller such as in [85] while some approaches include the uncertainties of the network as part of the model as in [86]. Gain-scheduling techniques [87] and model-predictive control techniques [88, 89] have also been proposed to handle network uncertainties in NCS.

The use of passivity-based techniques in NCS has been recently generating a lot interest because of the nice robustness properties that passivity provides. Passivity-based design of NCS is fundamental to our proposed contributions, therefore, in the Section 2.1.4 we review passivity-based NCS but before that we take a brief look at teleoperation, from which most of the passivity-based NCS designs are derived.

2.1.3 Teleoperation

Teleoperation, which naturally indicates to operate at a distance, extends the human capability to manipulate objects remotely by providing an operator with similar conditions as those in a remote environment [23]. The most common case of teleoperation is often associated with the control of a remote robot using a local robotic device, possibly a simple remote control. The main goals of teleoperation is to maintain stability of the overall system irrespective of the behavior of the environment and also to be able to provide the operator with a sense of telepresence, in the form of sensory feedback. Oftentimes in practice, the only feedback available to the operator of the local robotic device is visual. In most of the literature in teleoperation, various approaches are sought in order to design interfaces with the remote robot that provide rich forms of feedback in order to ensure telepresence and at the same time maintain stability. The underlying reasoning is that a high level of telepresence will provide the operator with a great sense of control of the remote robot, and therefore improve the precision and dexterity of the operator when performing tasks remotely. Applications of teleoperation are numerous, ranging from operating underwater vehicles [90], space robotics [91], telesurgery [92], operating mobile robots [93] and handling radioactive materials [94]. Most of these applications are safety-critical and hence effective and efficient strategies are of utmost importance in order to ensure that the two main objectives, stability and telepresence, are achieved.

Before the actual deployment of teleoperation, training is often required in order to get the operator accustomed to the task at hand. To facilitate training in teleoperation applications, *virtual environments* are often used. In this case, the remote robot is replaced by virtual interactions between objects in a simulated environment. The advantages of virtual environment has fostered research in such areas as virtual reality that are involved with the development of realistic virtual environments that assist in training potential operators in teleoperation [95].

One way to provide additional sensory feedback information is through *haptic technology*, or *haptics*. Haptics provide tactile feedback, and may be used when interfacing with a remote robot as in teleoperation, or with a virtual environment. In order for the haptic device to provide useful tactile information, it is important for the operator to "feel" in some sense the forces exerted on the remote robot (or within the virtual environment). This ability is termed *transparency*. One way to do this is by closing the loop on the forces acting on the remote robot (or actuate virtual forces from the virtual environment). Teleoperation with force feedback is known as *bilateral teleoperation*. The inspiration for most of the passivity-based techniques for NCS came from the field of bilateral teleoperaion. A bilateral teleoperation system, should be designed in such a way so as to provide the operator with realistic force feedback in a stable manner. However, bilateral teleoperation systems have an additional source of instability caused by the network through which the operator controls the remote robot. The network is a source of delays and data loss, which can destabilize even passive systems. Therefore, the wave variable formalism has been combined with passive techniques to address delays [96].

In a more recent work [97], performing bilateral teleoperation over the internet has been considered. In this work, it is shown that passivity in the communication channel can be maintained with both fixed and time-varying delays provided the controller handles dropped data and reordering of data appropriately. In [98], this idea is extended to both continous-time and discrete-time models of the teleoperation system by using the scattering transformation with communication management modules (CMMs). The CMMs, which are introduced to properly handle the wave variables output from the network so as to maintain passivity, consists of interpolation components and queue management. The interpolation components allow for compensation from packet loss and time varying delays. The queue management helps to mitigate bursty network behavior by compressing data during periods of decreasing delay and expanding the data during periods of increasing delays and packet loss. In this manner, the queue management prevents both extremes; namely, empty queues and queue overflow.

A very expressive approach that is well-suited for modeling teleoperation systems is the port-Hamiltonian formalism. *Port-Hamiltonian systems* are generalizations of port concepts from electrical network theory. Port-Hamiltonian systems are inherently passive and are easily augmented with wave variables [99]. Using the port-Hamiltonian formalism, the authors in [100] showed how to model a sampled data system. They showed that the interconnection between discrete and continuous port-Hamiltonian systems can be made to preserve passivity. An important factor for preserving passivity in a sampled data system is to ensure discrete energy leaps , normally introduced by the zero-order hold interface between the continuous and discrete time components, are handled properly. The developed port-Hamiltonian sampled data theory is then applied to telemanipulation and haptic interfaces to show that passivity is maintained even with dropped data and time-varying delays. This is done by construction rather than requiring the passivity observer and passivity controller as in [101].

2.1.4 Passivity-based Design of NCS

Passivity-based techniques have been successfully used in NCS for achieving robustness to network uncertainties [102] and provide significant advantages dealing with network delays, packet losses and quantization. Passivity properties has profound robustness to disturbance and uncertainties which makes it an important tool for NCS applications. Typically, passivity-based techniques used in NCS typically rely on the notion that all the components of the NCS are passive. In addition to that, information exchanged over the network is transmitted in such a way in order to preserve passivity. This is typically because network uncertainties can negatively impact the passivity of the overall NCS. For example, when delays are introduced in feedback loops such as in NCS, the network is no longer passive. One way to recover passivity or stability is through the use of scattering transformation or wave transformation. The work in [103] introduced the first use of scattering transformation in the setting of NCS. Using an NCS composed of a linear time-invariant (LTI) single-input, single-output (SISO) plant and controller in a feedback configuration, the authors showed that using the scattering transformation stability (asymptotic stability) of the closed loop system can be ensured independent of delays. An example application involving a a single degree of freedom pendulum was used to demonstrate the approach, the results showed desirable performance in the presence of time delays.

The passivity of the traditional feedback of two continuous-time systems in the presence of constant and time-varying delays in the feedback loop was explored in the work by Chopra in [104]. Using the notion of storage function as it relates to passivity, the authors show that with constant delays, the overall configuration is always passive. Additionally, in the presence of increasing delays, the authors showed that small gains bounded by the time derivative of the time-varying delays can be used to maintain a passive configuration.

In [105], the authors proposed a generalization of the scattering transformation for a special class of conic nonlinear systems [106], referred to as input-feedforward output-feedback-passive (IF-OFP) nonlinear systems. The objective of the approach is to maintain L_2-stability in the presence of arbitrarily large and unknown constant time delays. In the proposed methodology, the controller can be designed beforehand to meet very aggressive performance criteria under the assumption of no delays. Then the generalized transform is introduced to ensure stability of the system in the presence of constant delays and can be designed with low sensitivity to delay. By a comparison of their approach to the traditional small-gain approach and to the design without the generalized transform in simulations, the authors showed that their proposed approach demonstrated much better performance objectives while maintaining stability in the presence of constant delays.

Similar to the approach in [105], the authors in [107] proposed an approach to maintain passivity which utilizes a passive sample-data approach in the control of a continuous-time plant with a digital controller over a wireless network. Given that the information exchanged over a packet-switched network, the authors used the scattering transformation to produce continuous-time wave variables in order to send and receive information over a network. In order to maintain passivity, the sample-and-hold mechanism were performed using a passive sample-and-hold device to transform the wave variables into discrete-time. The passive sample-and-hold device used by the authors is based on ideas from the sampled data approach of [100]. As in [100], the approach maintains passivity even with certain types of time-varying delays and data loss. Further, the authors proposed a generalization of the approach to conic systems in multirate networks in the work [108].

In [109], introduced another approach to passivity-based NCS incorporating wave variables. This approach involves a component-based design of NCS in the presence of time-varying delays and data loss. The main objective of the work is the ability to control multiple components at the same time as well as the ability to add and remove elements (plants and controllers) from the NCS while maintaining passivity and L_2-stability (a form of bounded input, bounded output stability). The main component in the proposed approach is a passivity-based abstraction called power junction. The power junction, a hub-like component, interfaces between the plants and controllers within the star network. The power junction can be considered as a dissipative element and is essentially a wave digital filter [110], designed to guarantee passivity of the network. In regards to operation, the power junction takes as input wave variables and generates the same number of output wave variables in such a way that the input power is never less than the output power. The averaging power junction which averages the input power and generates all output waves equal to the average input power was introduced in [109] and is shown to maintain passivity in the network. It is also shown how to distribute the functionality of the power junction over all nodes in a ring network configuration. The work in [111] proposed an extension of this work, which provides an improved averaging power junction that exhibits better performance as demonstrated in the provided simulations. The work in [112] explored different implementations of power junctions. In this work, the authors used simulations to compared two lossless power junctions, the averaging power junction and a daisy-chain power junction, referred to as a consensus power junction. In [113], the authors introduced a distributed form of the power junction which is used in a multi-agent framework to perform various cooperative control objectives such as deployment. In this framework, each agent in the network has its own power junction, which is incorporated in order to ensure that the information exchanged between its neighboring agents maintains passivity and hence passivity of the overall networked system.

2.2 Adaptive Sampling

In most control applications, control engineers have typically designed controllers without considering that the required resources to achieve a particular control objective are limited. Specifically, in NCS, control engineers assume the channels between sensors, controllers and actuators are infinite bandwidth, noise free and delay-free. As have been witnessed over the years, the effects of limitations and uncertainties of the communication network can no longer be neglected. Owing to *the limited resources*, the fundamental questions in designing control systems over a communication network are "*how frequent should a system be sampled*" and "*how often should information be sent over a communication network*". The classical and well-studied approach is the periodic sampling approach, whereby a system is sampled based on a fixed interval that is set a-priori. In this approach, engineers tend to rely mostly on rules of thumb such as sampling with a frequency of between 20-40 times the system bandwidth[114]. This approach is typically referred to as an "*open loop sampling approach*" because regardless of the condition of the overall system, the sampling interval is fixed. A recent effort has brought about a shift in perspective by *event-triggered control* [115, 116], which is a type of the generalized notion of *adaptive sampling*. In this paradigm, rather than the periodic communication and computation of control law and sensor signals, sampling is the based on the notion that the occurrence of an event results in the control and/or communication of control law updates and sensor signals.

Adaptive sampling can generally be categorized based on the two main NCS components from which an event or triggering behavior for adaptation originates. The two categories are:

2.2.1 Event-Triggered and Self-triggered Control

In *event-triggered control*, the sensor transmissions and control updates are initiated at instants defined by the occurrence of events taking place at the sensor or controller. Typically, the events are generated when a certain triggering condition defined as a function of system state exceeds a certain threshold. The concept of event-triggered control has appeared under different names such as interrupt-based feedback [117], Lebesgue sampling [115] or state-triggered feedback [118]. In event-triggered control, a hardware event detector is typically required to generate a hardware interrupt in order to signal the violation of a condition and the initiation of sampling. This is typically done using either field-programmable gate array (FPGA) processors or application-specific integrated circuit (ASIC). Hence, provided the cost associated with the hardware detector is reasonable, the event-triggered control is a very useful method for adaptive sampling. It provides the benefit of reducing communication transmissions and also the energy consumptions of computing and sensor nodes potentially extending the battery life span of these systems.

In the past few years, there has been numerous efforts in the design of event-triggered control approaches. The author in [119] used input-to-state (ISS) stability property of a system with respect to measurement errors to derive an ISS event-triggering scheme which guarantees that the Lyapunov function of the system is monotonically decreasing. This triggering scheme was extended in [120], and with the new scheme the authors noted that the monotonically decreasing requirement in [119] is not necessary to guarantee asymptotic stability at each sampling/transmission instant. Other lyapunov-based approaches were proposed in [121]. Dead-band approaches, whereby sampling or transmission is triggered when a measurement exceeds a threshold, were utilized in [122, 123, 124, 125, 126]. In [127, 128, 129], event-based control techniques for linear stochastic systems were proposed. A passivity-based approach for ensuring L_2-stability in the event-triggered framework was explored in [130].

In [131, 132], the authors explored the application of event-triggered control to multi-agent systems. Other works investigated distributed event-triggered control [133, 120]. The authors in [134, 135] proposed an architecture for performing event-triggered control for unknown or uncertain plant models.

Another well-known adaptive sampling control approach is *self-triggered control*. Self-triggered control is an alternative to the event-triggered control especially in the cases where it may be impractical or unreasonable to integrate event detectors. In contrast to the event-triggered control, self-triggered control is a software-based approach. In self-triggered control, the next sampling or triggering instant is computed based on the current measurement and the model of the system. This can be seen as an emulation of the event-triggered scheme described above. The benefit of this approach is the lack of the need for continuously monitoring a triggering condition. Self-triggered control was introduced in [136] in which a heuristic approach is used to adjust sampling intervals. Several research studies have been proposed for this approach both for linear and nonlinear systems [137, 138, 139].

Minimum-attention control, another adaptive sampling approach, which was introduced by [140] has been recently revisited in [141, 142] as well as *anytime control* [143, 144], with the overall objective of maximizing the sampling interval when performing closed-loop control over networks by essentially minimizing the attention the control loop requires, while guaranteeing a certain level of closed-loop performance. This approach is similar to the self-triggered control but the difference lies in the fact that unlike the self-triggered approach, the minimum-attention control approach is typically not designed using emulation-based approaches in the sense that it does not require a separate feedback controller to be available before the triggering mechanism can be designed.

In most of these works that are devoted to the development of adaptive sampling techniques for control systems, the interaction with the network has not been fully addressed. Only a few of the approaches have attempt to address the impact of network effects such as packet losses and delays. An analysis of event-triggered control with packet losses and delays has been presented in [145, 146, 147, 148] for simple wireless network abstractions. The system-level design of self-triggered controllers over a wireless network for a single control loop and multiple loops has been addressed in [149].

2.2.2 Network-based or QoS-based adaptive sampling

In *network-based* or *QoS-based sampling*, the sampling rate of the NCS is determined based on network quality of service (QoS) metrics such as delays, throughput and frame loss. In this approach, adaptive sampling is triggered by the response to events or activities from the network component of

NCS and hence determines when to send sensor updates as well as control updates over the network.

Various research efforts have been devoted to sampling rate adaptation, as it is commonly referred to, from the network perspective. The work in [150], motivated by the goal of maintaining low levels of data loss while keeping to the highest possible rate of sampling and control, showed positive results have been obtained by dynamically adjusting sampling rates to reflect changing network conditions. [151] proposes an heuristic algorithm to adapt bandwidth allocation of control systems over a CAN bus based on two factors, network load and stability threshold. Another heuristic approach was provided in [152], where a heuristic approach was proposed to adaptively adjust the sampling rate of NCS based on QoS metrics, delay and packet loss, for the overall objective of improving the quality of control performance. In [153], control systems vary their sampling periods based on the congestion level of Wide Area Networks (WANs) which is fed back from the network. It allocates bandwidth to avoid network congestion of WANs and preserves high performance level for NCS. A convex optimization is lower bounded by the minimum rates that guarantee system stability, and upper bounded by the total network capacity. [154] varies the sampling rate period of each controller based on the state estimation of network conditions, system stability/performance requirements, and computation/bandwidth limits of the hardware. It is then solved by dual Lagrange multipliers in a fully distributed manner. The work in [155] adapts the sampling interval based on the measurement of round-trip delay and assures stability in the mean square sense using discrete-time Markov jump linear system (MJLS) theory. The MJLS is based on an ‘a priori’, static sampling policy, with network dynamics described as linear time-invariant systems switching between a finite combination of sampling interval and delay. Bao [156] proposed a rate allocation technique to minimize the distortion introduced by quantization over a noisy channel. An optimization problem is constructed with the objective of minimizing the linear quadratic cost by means of mean squared error (MSE), constrained by the total rate, and solved using Lagrange duality. Ploplys in [157] presented experimental results on the NCS over a wireless network whereby changes in the network performance resulted in the adaptive tuning of the sampling rate.

2.3 Model-Based Design and Simulation of NCS

In this section we provide a brief introduction of model-based design in control systems as it relates to NCS. We also describe the various approaches for the simulation of NCS.

2.3.1 Introduction of Model-Based Design in Control Systems

Model-based design methodologies are emerging as the most convincing alternatives in addressing the difficulties and complexities involved with the development of embedded control systems compared to the classical methods based on low-level languages and intense prototyping activities [158, 159]. Model-based design for embedded control systems involves creating models and checking correctness at different stages in the development process [160]. The design flow progresses along precisely defined abstraction layers and typically follows the well-known “V-model”. In the model-based design process, there is a defined hierarchy of models representing the various design decisions. Typically, the design starts with an abstract model of the system based on coarse specifications. Subsequently, the designers refine the design and the models of the subsystems and components until the desired level of detail that accurately represents the dynamics of the component under consideration. For control systems, it starts with control design and then proceeds to simulation and debugging with dedicated tools, followed by system-level design for the specification of platform details, code organization, and deployment details, and the final stage of integration and testing on the deployed system.

2.3.2 Model-Based Design of Networked Control Systems

The model-based design for embedded control systems, described above, cannot be directly applied to NCS because the domain heterogeneity and the existing tight coupling between design concerns in NCS create a number of challenges. Ensuring controller stability and performance for physical systems in the presence of network uncertainties (e.g. time delay, packet loss) couples the control and system-level design layers. In addition, downstream code modifications during testing and debugging invalidate results from earlier design-time analysis and any component change often results in restarting the design process.

A number of research projects seek to address the problems of model-based design for NCS. The ESMoL modeling language for designing and deploying time-triggered control systems explicitly captures in model structure many of the essential relationships in an embedded design [161, 162]. The ESMoL tools include schedule determination for time-triggered communications, code generation, and a portable time-triggered virtual machine. AADL [163] is a textual language and standard for specifying deployments of control system designs in data networks [164]. AADL projects also include integration with verification and scheduling analysis tools. The Metropolis modeling framework [165] aims to give designers tools to create verifiable system models. Metropolis integrates with SystemC, the SPIN model-checking tool, and other tools for scheduling and timing analysis.

2.3.3 Simulation of Networked Control Systems

As NCS become increasingly complex, it becomes more challenging to formally analyze NCS properties such as stability and performance. As a result, there is a pressing need to evaluate both the control system and the networking system holistically for a rapidly growing number of applications. Simulation is a powerful technique for evaluation and can be used at various design stages.

An intuitive approach is to engineer a new tool from scratch that would essentially contain modules of the control system dynamics and for the communication network but a good practice and basic principles of engineering is to rely on well-developed ideas as much as possible and avoid reinventing the wheel [166]. The approach of engineering a new tool has been explored in [167, 168, 169]. However, the use of existing tools seems like the right approach based on basic engineering practices. There are two main approaches in using existing tools for the simulation of NCS. The first approach involves extending the features of an independent simulator to enable the simulation of both the control dynamics and communication network of a NCS, we refer to this category as *monolithic frameworks*. The second approach involves the integration of independent tools whereby each tool is designed for simulating a specific domain (control dynamics or the communication network) of a NCS, we refer to this category as *heterogeneous simulation frameworks*.

2.3.3.1 Monolithic Simulation Frameworks

The approach involving the extension of independent simulators for NCS simulation can be further categorized into two approaches, the *extension of network simulators* and the *extension of simulators for dynamical systems*.

1. Extending Network Simulators

This approach involves extending independent network simulators to support the simulation of physical systems and control dynamics. Various attempts have been made to extend network simulators such as ns-2 [170] and OMNet++ [171] to simulate NCS. In [169], the authors extended ns-2 in order to simulate the dynamics of plants and controllers that are modeled by differential equations (solved via a linked package), this approach was later extended in [172].

The main issue with this approach is that it requires the physical system and control algorithm to be fully implemented in a high-level language such as C++. This becomes very difficult as the complexity of the NCS increases. Hence, using this approach is very difficult to accurately model and simulate the dynamical systems of NCS.

2. Extending Simulators for Dynamical Systems

This approach mainly consists of extending simulators for dynamical systems to also simulate the events and dynamics of the communication network. Various attempts have been made towards this direction to extend simulators such as Matlab/Simulink [173], Modelica [174] and Ptolemy [175] etc. For example, network simulation is provided in Matlab/Simulink using add-ons such as TrueTime [167]. A new version of TrueTime has also been developed to extend Modelica to enable the simulation of communication networks [176]. VisualSense is an extension of Ptolemy that enables the simulation of the communication network [177].

The main issue with this approach is in regards to the accuracy of the simulation of the communication network which depends on the level of abstraction of the network protocol models. For example in TrueTime, the network protocol only supports the simulation of the medium-access and physical layers but not higher level protocols such as TCP or UDP protocols, which are essential to precisely model the network stack in order to simulate the communication network of a NCS. Hence, these tools lack the fine-grained simulations of network dynamics [166].

Based on the two monolithic simulation frameworks, it can be seen that in order to develop a realistic and accurate simulation of NCS, it would be best to integrate existing tools for the accurate simulation of the control dynamics as well as the networking system of a NCS.

2.3.3.2 Heterogeneous Simulation Frameworks

Heterogeneous simulation frameworks coordinate the simulation of multiple different types of models as opposed to a monolithic framework. The heterogeneous models may typically be composed of various aspects of a single system or of distinct systems within a system. This framework is ideal for the simulation of NCS since various tools can be integrated in order to simulate the dynamics of NCS. The integration of existing tools for the accurate simulation of NCS, although very beneficial, faces several challenges such as the time synchronization between the simulators which is critical to preserve the correctness of the simulation, the data communication between the simulators to ensure consistent data semantics during the simulation and scalability of modeling NCS in the framework.

Several efforts have been made toward integrating multiple simulators in order to effectively simulate NCS. PiccSIM presented in [178] allows the integration of Matlab/Simulink models with ns-2 and provides a graphical user interface for the design of NCS and the automatic code generation of ns-2 and Matlab/Simulink models. In [179], a special simulator interface, implemented in C/C++ is used to integrate the simulators, ModelSim, Matlab/Simulink and ns-2 to establish the communication between the simulators. Other integration tools for NCS include [180][181], ADEVS/ns-2 integrated tool [168], Modelica/ns-2 integrated tool [182] which was extended in [166] to provide a more accurate synchronization and ability to handle event-triggered control systems.

The above approaches described for the integration of simulators are not standardized frameworks and hence it is difficult to evaluate the accuracy of the integration. An example of such a standardized framework for integration of simulations is the *High Level Architecture* which we will review in the following section.

2.3.3.3 High Level Architecture

The *High Level Architecture* (HLA), originally defined by the Defense Modeling and Simulation office for the U.S. Department of Defense, is a standardized software architecture that provides the ability to link different kinds of simulations, simulators, models and other tools, each designed for a specific problem domain. The primary motivation for its development was to support "composable" simulation [183]. HLA comprises of three main components the Federation framework and rules, the Run-Time Infrastructure (RTI) Interface specifications, and the Object Model Template. In HLA, the independent simulations, simulators, models and tools are known as the *federates*. An HLA simulation composed of the federates is known as the *federation*. The RTI is the software component that implements the HLA interface specification and runs the federation. It provides a set of services to support federate to federate interactions and federation management support functions. Communications between different federates is based on shared *objects* and *interactions* whereby objects are analogous to the shared memory in an operating system and are owned by one federate while interactions represent message passing.

The HLA RTI interface specification describes six types of runtime services between federates and the RTI, they include *Federation Management*, *Declaration Management*, *Ownership Management*, *Object Management*, *Time Management* and *Data Distribution Management*. The *Federation Management* service provides functionality that allows for creation of federations, joining and resigning of federates and the overall management and synchronization of federations. The *Declaration Management* service provides for an efficient data exchange between federates. Each federate use this service to define which data it will provide to and require from the federation. Through the *Ownership Management* service, RTI controls ownership of the HLA objects and their attributes. Only a federate owning the attribute can update the attribute of an HLA object in a federation. However, RTI can dynamically transfer the ownership of an attribute/object during a federation. The *Object Management* service facilitates the creation, deletion, identification and other services at the object level. The *Time Management* service is responsible for synchronization of run time data exchange. Through this service, RTI controls when federates update and receive the events. Finally, the *Data Distribution Management* services allows for efficient routing of data among federates [183, 184].

The HLA OMT is a standard template that defines form, type and structure of data shared within the federation. The OMT defines two formats for describing this information. HLA specifies two types of object models namely Federation Object Model (FOM) and Simulation Object Model (SOM) and requires that these models be documented in accordance with the OMT. The FOM describes the set of objects, attributes and interactions that are shared across a federation. The SOM describes the simulation (federates) in terms of the types of objects, attributes and interactions it can offer to future simulations. A federation FOM is composed of parts of the SOM of all of its participating federates. All the federation configuration information are stored in a standardized format text file called the *FED* file [183, 184].

HLA provides a flexible control of time in an integrated simulation. In the HLA framework, each federate must maintain a clock corresponding to the internal logical time of its simulation which is distinct from any real-world clock. HLA provides numerous schemes for coordinating the evolution of the logical clock among federates and can range from completely lacking time synchronization, where one federate can execute arbitrarily far into the future, to completely synchronized, where all federates evolve time within a tightly bound window. A federate can be configured to be time-constrained or time-regulating, both, or neither [184]. A time-regulating federate's progression constrains all time-constrained federates. Likewise, a time-constrained federate's advance is controlled by all time-regulating federates. A federate that is neither constrained or regulating is free to evolve time on its own. The times of the federates that are both evolve in tight lockstep. If, for example, all federates can run at least as fast as real-time, and one federate tightly correlates its time advance requests to wall-clock time, then the entire federation can be made to run in real-time. Otherwise, it is possible to execute simulations both faster and slower than real-time.

To operate in the HLA framework, each federate typically defines a step size, look-ahead interval and minimum time stamp. When a federate requests to evolve its internal simulation time it does so in increments of step size, which may vary in size from step to step. The look-ahead corresponds to the amount of time into the future which the federate guarantees it will not issue an interaction or object update and is generally small compared to step size. When the federate is in a Time Advance Request (TAR) state, minimum time stamp is defined as the federate's requested time plus look-ahead. When the federate is in a Time Advance Granted (TAG) state, minimum time stamp is the federate's logical clock time plus look-ahead. Each federate maintains an understanding of all other federates' minimum time stamps. A similar protocol is supported for event driven simulation in which the event driven simulation requests the next event from the RTI. The simulation logical time is advanced either to the earliest available interaction or to the time stamp of the next event local to the requesting simulation [184].

In the HLA framework, each of the simulation engine and models must be individually integrated at both the HLA API level and at overall simulation level in order to participate in the federation. Although, HLA APIs provide run-time support, the integration of models in the environment is not addressed. Recent efforts in the Command and Control Wind Tunnel (C2WT) project have been explored in order to address this challenge. The C2WT is a robust multi-model simulation framework for integrating heterogeneous simulation components using the RTI within the HLA platform [185, 186]. It captures not only the necessary interface specification for running heterogeneous simulations over HLA, but also a variety of mechanisms for configuring, enhancing, and detailing the simulation execution. The C2WT framework uses the discrete event model of computation as the common semantic framework for the precise integration of an extensible range of simulation engines. Formal semantics of heterogeneous models are difficult to capture because the component models may be defined using dramatically different domain specific modeling languages (DSMLs). To integrate the models, the C2WT uses meta-modeling and the meta programmable Model Integrated Computing (MIC) tool suite [187] for developing a Model Integration Layer [186], which can be used to formally define the simulation semantics. The tool suite facilitates the generation of over-arching run-time software components and glue code necessary for the execution of a HLA-based simulation.

2.3.4 Functional Mock-Up Interface (FMI) and Functional Mock-Up Unit (FMU)

The growing trend in system design involves collaborative efforts whereby different groups design and test various components of a system independently and at the very end all the components are then integrated to get the complete system. With this trend, there is a strong need for the effective exchange of models between the various groups to facilitate effective system development. Additionally, tools used by each group might be different. Hence, there's a strong need for tool independent standards for model exchange as well as co-simulation in order to evaluate various system properties. Recently, a standardization effort was started to facilitate co-simulation and integration of models into different simulation environments and execution of models on embedded systems. The *Functional Mock-Up Interface* (FMI), first initiated by Daimler AG, is a tool independent standard introduced for the exchange of dynamic models and for co-simulation. The primary objective of the FMI is to support the exchange of models between suppliers and original equipment manufacturers (OEMs) even if a large variety of different tools are used. FMI standard consists of two main parts, *FMI for Model Exchange* and *FMI for Co-Simulation* [188].

1. FMI for Model Exchange: The main objective of FMI for Model Exchange is allow any modeling tool to generate C code or binaries representing a model which can then be easily integrated into another simulation environment.

2. FMI for Co-Simulation: This interface standard provides a solution for co-simulation of time dependent coupled systems consisting of subsystems that are continuous in time (model components that are described by differential equations) or discrete in time (model components that are described by difference equations like, e.g. discrete controllers). The standard defines interface routines for the communication between a master and the individual simulation

tools (slaves) in the co-simulation environment. The data exchange is restricted to discrete communication points in time and the subsystems are solved independently between these communication points.

The distribution of FMI is enabled by a component which implements the FMI called *Functional Mockup Unit* (FMU). It consists of one zip-file with extension ".fmu" containing all the necessary components to utilize the FMU. The included components consists of the following:

- XML-file: This file contains the definition of all variables of the FMU that are exposed to the environment in which the FMU is used, as well as other model information. It is then possible to run the FMU on a target system without unnecessary overhead. For FMI-for-Co-Simulation, all information about the "slaves", which is relevant for the communication in the co-simulation environment is provided in a slave specific XML-file. This includes a set of capability flags to characterize the ability of the slave to support advanced master algorithms, e.g. the usage of variable communication step sizes, higher order signal extrapolation etc.

- C-functions: For the FMI-for-Model-Exchange case, all the needed model equations are provided with a small set of easy to use C-functions. These C-functions can either be provided in source and/or binary form. Binary forms for different platforms can be included in the same model zip-file. For the FMI-for-Co-Simulation case, a small set of easy to use C-functions are provided in source and/or binary form to initiate a communication with a simulation tool, to compute a communication time step, and to perform the data exchange at the communication points.

- Additional data: Further data can be included in the FMU zip-file, especially a model icon (bitmap file), documentation files, maps and tables needed by the model, and/or all object libraries or DLLs that are utilized.

2.4 Dependability of Networked Control Systems

Parallel to the increased complexity of NCS, there is an increasing demand for dependable NCS which has led to the emergence of research efforts in fault-tolerance control and monitoring systems in NCS. NCS depends strongly on the availability and correct functioning of both the control system components as well as the communication network. The occurrence of a fault can trigger a system failure that can possibly lead to catastrophic consequences. Hence, a critical issue in the design of NCS is robustness or tolerance of the system with respect to faults including system component failures as well as network failures. By network failure, we mean a total breakdown in the communication between the control system components as a result of, for example, some physical malfunction in the networking devices or severe overloading of the network resources that cause a network shut down. Therefore, in the presence of faults in NCS, dependability becomes a very important issue.

In this section, we provide a brief overview of dependability and a brief introduction to fault-tolerant control. We then focus more on a form of reconfiguration in active fault-tolerance, analytical redundancy, as it is most related to the proposed research.

2.4.1 Dependability

The concept of *dependability* of a system is defined as the quality of a system behavior as perceived by another interacting system such that reliance can be justifiably placed on the system's behavior [189]. Based on the survey paper on the fundamental concepts of dependability in [190], dependability consists of three parts:

1. The ***threats*** to the correct functioning of the system, such as faults, errors and failures.
2. The ***attributes*** encompassing dependability which include:
    - *availability*, which indicates the readiness for correct system behavior.
    - *reliability*, which means the continuity of correct system behavior.
    - *safety*, which denotes the absence of catastrophic consequences on the users and the environment.
    - *confidentiality*, indicates absence of unauthorized disclosure of information.
    - *integrity*, which represents the absence of improper system state.
    - *maintainability*, which indicates the ability to undergo repais and modifications.
3. The ***means*** of achieving a dependable system include:

- *fault prevention* involves methods and how to prevent the occurence of faults.
- *fault tolerance* involves approaches on how to provide desirable system behavior in spite of the occurence of faults.
- *fault removal* entails methods on how to minimize, through the process of verification and validation, the presence of latent faults.
- *fault forecasting* involves techniques for estimating through thorough evaluation, the presence, the creation and consequences of faults in a system.

Closely related to our proposed work is the use of fault-tolerance to ensure reliability of NCS in the presence of faults. Along those lines, we introduce the notion of fault-tolearance.

2.4.2 Introduction to Fault-Tolerance

In a general sense, a fault is an event that changes the behavior of a system such that a system no longer satisfies its objective or purpose. There exists numerous approaches of categorizing faults based on the system. A detailed survey paper in [190], described six categories of faults namely: phenomenological cause, intent, phase of creation or occurrence, domain, system boundaries and persistence. The authors in [191] classified faults in a typical control system as follows:

- *Component faults*: These comprise of faults that change the dynamical input-output properties of the system.
- *Sensor faults*: These faults occur when the sensor readings have substantial errors, but the plant properties are not affected. Common types include bias, drifts and calibration errors of sensors.
- *Actuator faults*: These faults occur when the influence of the controller on the plant is interrupted or modified but the plant properties are not affected.

Faults in dynamical systems can, in general, be categorized using the criteria based on the *nature of the fault* or the *persistence of the fault* as follows:

1. **Nature of Fault**: These include
    - *semantic or value fault* occurs when an incorrect value is provided to a system.

- *timing fault* occurs when the value is presented outside the specified time interval.

2. **Persistence of Fault**: These include

   - *transient fault* is a fault that exists only for a short period of time and disappears by itself.
   - *permanent fault* is a fault that exists until an explicit repair action takes place.

Based on the persistence classification of faults, the authors in [192] suggested that a lost measurement can be treated as a transient fault in the sensor and a lost command can be treated as a transient fault in the actuator. Thus disturbances/faults in the communication network can be masked as disturbances/faults in the control system and therefore strategies in the Fault tolerant control (FTC) field can be applied to NCS which can handle the above situations and allow the evaluation of stability and performance of the control system.

Generally, in order to identify the faults of a system, a diagnosis component is necessary in order to identify the source or cause of the fault. The development of diagnosis classes is done by making use of previous experience, knowledge or information within an application area. The necessary information used may come from several sources of knowledge, such as from system analysis. [193] provides a survey on fault diagonsis, in particular for dynamical systems.

The general approach for making a system fault-tolerant consists of two steps:

1. **Fault diagonsis**: This step involves determining if a fault has occurred (*fault detection*), then finding out where the fault has occured (*fault isolation*) and then figuring out the level or impact of the fault (*fault estimation*).

2. **Reconfiguration**: This involves adapting the control system to the faulty situation so that the overall system continues to satisfy its objective possibly with a possible decrease in performance. Reconfiguration could involve closing the control loop via redundant signal path. The redundant signal path can be an implicit property of the communication network employed within the control system, and can be optimally exploited in the communication protocol.

These steps are typically designed in an integrated way, so that the reliability of the fault-tolerance scheme is high.

2.4.3 Fault-Tolerance Control (FTC)

Fault-tolerant systems and associated control designs have inherently wide and diverse engineering applications, and can be divided into four main categories as follows [194]

- *safety-critical systems* such as aircraft, helicopters, spacecraft and automobiles, nuclear power and hazardous chemical plants;
- *life-critical systems* such as tele-robots for surgery, implanted heart monitors, nanoscale diagnostic instruments, digital protheses and other medical devices, as well as ground traffic control and automated highway systems;
- *mission-critical systems* such as avionics and air traffic control systems, defense systems, spacecraft and space stations, autonomous aerial/space/underwater vehicles, robots used in industrial processes, and communication networks;
- *cost-critical systems* such as large-scale space structures, drive-by-wire automobiles, distributed process control, computing and communication networks.

FTC is a very active research area and describes techniques for adaptively adjusting control loops to faulty plants. An FTC system has the property that faults do not develop into a failure of the closed-loop system. The system is said to be *fail-operational* when it remains in operation after the occurrence of a fault and its performance stays the same. When its performance decreases, the system is called *fail-graceful or fail-passive* [195]. If a fault triggers a situation when cannot be kept up, the system is brought into a safe state and switched off, this system is said to be a *fail-safe system*.

FTC techniques can be divided into two main catergories, *passive FTC* and *active FTC* as shown in Fig. 3 [195]

1. **Passive FTC** [195] In *passive FTC*, the conceivable system component failure is assumed to be known a-priori, and the control system takes all failure modes into account in the design stage. Once the control system is designed it will remain fixed during the entire operation. Hence, passive FTC deals with a presumed set of system component failures which are considered at the controller design stage. Robust control is a type of Passive FTC, used in

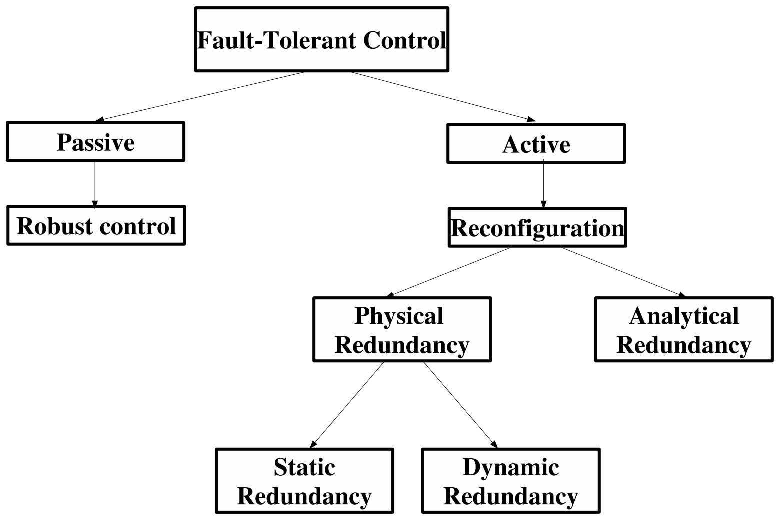


Figure 3: Classification of Fault-Tolerant Control

designing a controller to tolerate a set of possible faults. In this case, the control system is considered to be *robust* because even in the event of component failures, the control system should still be able to maintain the desired level of performance. However, the set of faults that can be tolerated without active controller re-adjustment is usually limited. Also, for the designed controller to maintain a certain level of performance under a wide range of failures, the designed controller is typically conservative. The drawback of this approach is that one has to ensure the control system works under all possible system operating scenarios and in the presencce of an unaccounted failure nothing can be said about the behavior of the system. Hence it is very difficult for a passive FTC to be optimal from a performance standpoint alone.

2. **Active FTC** [195] In contrast to passive FTC, *active FTC* represents techniques to achieve fault tolerance by changing the control loop after a fault has occurred in order for the stability and performance of the overall system to be maintained at an acceptable level. This approach relies heavily on the ability to diagnose fault for the most update information about the system and operating conditions of the components. The fault diagnosis step typically involves Fault Detection and Isolation (FDI) which seeks to find out whether the system is subjected to a fault and to identify the fault. The fault diagnosis step is then followed by a controller adjustment step, called control reconfiguration. The main critical issue facing any active FTC is the limited reaction time available to detect, diagnose and reconfigure control actions in

reaction to faults. The speed, accuracy and robustness of these operations are the main factors to the success of any active FTC.

Active FTC can be categorized into *Physical Redundancy* and *Analytic redundancy*. *Physical redundancy* includes conventional fault-tolerant control systems that are achieved and ensured through hardware redundancy, that is, by including redundant actuators and sensors in the system. The control and measurement channels are generally made duplicated or triplicated in hardware. The main disadvantage of physical redundancy is the additional cost and the corresponding increase in complexity of operation. Moreover, the weight of the system and the maintenance requirements are subsequently increased. The two types of physical redundancy are *static redundancy* and *dynamic redundancy*[196]. In static redundancy, typically three or more parallel modules are used that have the same input signal and are all active. A voter compares their output signals and decides by majority which is the correct one. In this scheme for example in a commonly used triple-redundant modular architecture one fault can be masked without the use of special error detection methods. On the hand, *Dynamic redundancy* requires fewer modules at the cost of more information processing. A minimal configuration consists of two modules, where one module is in operation, and the second module takes over in case of an error. When the second module is continuously operating this method is called hot standby, which has the advantage of shorter downtime of the system. However since it is operating all the time aging of the module becomes a disadvantage. This can be circumvented using a cold-standby configuration, where the backup system is normally out of function and only becomes operational in case of an erroneous primary system.

Contrary to the physical redundancy, *Analytic redundancy* is a technique that replaces the hardware redundancy by a process model which is implemented in the software form and hence sometimes referred to as *software redundancy*. The next section will provide a review of analytic redundancy.

2.4.3.1 Analytic Redundancy

The concept of *Analytic redundancy* or so called *model-based FTC* refers to redundant components in a system with different specifications, designs and implementations, but satisfaction of certain requirements that are analytically related [197]. The first application of analytic redundancy for software fault tolerance in control systems was introduced by [198]. In analytic redundancy, an explicit mathematical model is used to perform two steps. The fault is diagnosed by using the information included in the model and in the on-line measurements from the sensors. Then the model is adapted to the faulty situation and the controller is re-designed so that the closed-loop system including the faulty plant satisfies the given specifications. In this case, a model is essentially a quantitative or a qualitative description of the system dynamic and steady behavior, which can be obtained using well-established system modeling techniques. In this way, the system behavior can be reconstructed on-line [199]. In other words, redundant system models will run in parallel to the real process and it will be driven by the same inputs.

Analytic redundancy, making use of mathematical model of the system and relationships between sensor outputs and actuator inputs, has been proposed and are increasingly being employed in complex control systems [200, 201]. A very well known form of analytic redundancy is the *Simplex Architechture* introduced by Sha in [202, 203, 204], an architecture that provides safety guarantees by “using simplicity to control complexity”.

2.4.3.2 Simplex Architecture

In [202, 203, 204], the original *Simplex Architecture* provides a fail-operational mechanism for a malfunctioning software controller. The architecture permits online modification and upgrade to control software without sacrificing safety. It involves a robust design where complex controller faults can be detected and fixed during run-time without jeopardizing safety. The Simplex Architecture uses three subsystems: safety, complex, and decision. The safety subsystem has a simple, reliable controller which provides verifiably safe performance and is used in case the complex controller malfunctions. The complex subsystem drives the system during regular operation, as long as it does not jeopardize system liveliness. This controller can be changed and upgraded while the system is running and may even contain bugs (therefore it does not need to be verified, or may even be too complex to fully verify). The decision subsystem chooses which of the two previously-mentioned controllers to use.

The work in [202, 203, 204] developed the Application-Level Simplex Architecture which has all three subsystems located at the application-level. This works well for protecting the system from faults directly from complex controller, however it does not provide safety for indirect faults of underlying components. For example, if the operating system which runs all three subsystems in the Application-Level Simplex Architecture contains a bug and crashes, plant safety can no longer be guaranteed. An extension to Application-Level Simplex Architecture was provided in [205] which uses hardware/software codesign to also tolerate software faults in other dependent layers such as microprocessor and operating system.

Another important software fault-tolerant approach is N-version programming [206]. In this method, multiple versions of software are independently created from the same specification. Then, all are run and the result given by the majority of versions is taken as the output of the system. One drawback with this method is the lack of statistical independence of faults [207]. Additionally, for a constant amount of development effort, N-version programming is actually less reliable than focusing on a single version over a wide range of parameter values [204].

The recovery block approach is another approach similar to the simplex architecture [208]. In this approach, several alternative methods are developed. The first fully feature method is ran first and checked for correctness, if it is, this method is used. Otherwise, a simpler method is attempted. The clear difference between the recovery blocks and the Simplex Architecture is that the the recovery block is a backward recovery method while the Simplex Architecture is more of a forward recovery method.

2.4.4 Model-based Detection

Model-based detection has a long rich history finding immense application in fault diagnosis. A survey of the various detection, isolation and recovery approaches is provided in [209]. In [210], the authors survey various quantitative and qualitative model-based approach used in diagnosis. We focus on the quantitative model-based approaches as it is directly related to the work presented in this manuscript. In model-based detection, the underlying objective is the determine using a model of system whether the system is exhibiting a nominal or abnormal behavior. The system model is usually developed based on some fundamental understanding of physics of the system. The system's model can be an analytical model represented by a set of differential/difference equations or it could also be a knowledge-based model represented by, for example, neural networks, fuzzy rules or expert knowledge [211]. Based on the system model and corresponding inputs to the system, the behavior of a system can thus be predicted online. The underlying idea then becomes to compare the difference between the predicted and actual system behaviorsx. The differences in behavior are typically known as residuals. The system is said to behave nominally if the residuals are within predetermined or specified bounds and in the case when the residuals exceed the specified thresholds the system is said to behave abnormally. The first step in model-based detection, which involves the process of creating the residuals, is called residual generation. The second process in model-based detection, which involves the extracting and evaluating information from the generated residuals, is called residual evaluation. These two processes are at the core of model-based detection. Figure 4

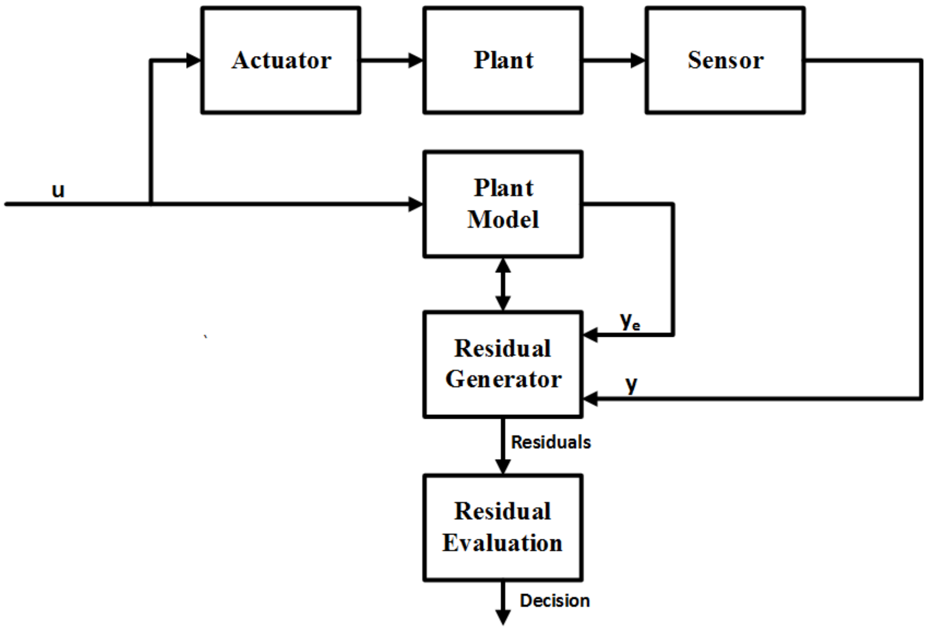


Figure 4: Model-Based Detection Scheme

shows the typical model-based detection scheme, which is composed of the plant model, the residual generation and residual evaluation components. In what follows we further describe the residual generation and evaluation components along with the existing approaches in the area.

2.4.4.1 Residual Generation

Residual generation, shown in Figure 4, is particularly useful especially in the presence of model uncertainties, measurement and process noise and unknown disturbances. The most frequently used residual generation techniques are discussed as follows.

1. **Observer-based residual generation:** This is the most common type of residual generation technique, and essentially involves the use of an observer or a bank of observers in order to estimate the output of the monitored system, subsequently the estimate is compared to the actual system output in order to obtain the residual [212]. The observer-based method is applicable to wide range of systems including static systems, dynamic systems, stochastic systems etc. Contrary to the classical observer used in the control community, observers used in residual generation, typically referred to as diagnostic observers, are essentially output observers that estimate the measured states of the system. Notwithstanding, full state observers are also used in residual generation and provides the additional degree of freedom which aids in the isolation process [213].

2. **Parity space based residual generation:** In these approaches a parity relation is derived from the system model. Parity equations are typically obtained by transforming or rearranging the system's input-output models. Using online system data these models are relatively easy to generate and use. [214, 215]. The main objective is to check the consistency or parity of the system model with measured outputs and known inputs. This approach was first proposed in [214] for state space representation of a system and subsequently extended to system models represented as transfer functions [216, 217]. The work in [218], showed that the parity space approach results in certain types of observer structures. Due to this fact, even though the design techniques are different, the parity approach is said to be structurally equivalent to the observer-based approach.

3. **Parameter identification:** The foundation of the parameter identification based methods is the on-line estimation of parameter(s). This approach requires accurate parametric model of the monitored system [219]. The benefit of this approach lies in the fact that it yields the size of deviation of parameter and this has be found to very useful in particular in the process of

analyzing faults. The main drawback of this approach is that in order for parameters to be accurately estimated , sufficient excitation, which is not always available, is needed. [211]. Some of the existing parameter identification schemes, just to name a few, include techniques based on least squares, recursive least squares, extended least squares etc.

2.4.4.2 Residual Evaluation

Residual evaluation, shown in Figure 4, is a very important component of the model-based detection scheme. This component essentially decides whether or not an abnormal behavior is present based on the obtained residuals. The evaluation schemes can be categorized into schemes for stochastic and deterministic systems. For stochastic systems, the residuals can be evaluated using statistical or stochastic properties such as mean, variance likelihood ratio (LR), generalized likelihood ratio (GLR) [220], sequential probability ratio testing, CUSUM etc [209]. On the other hand, for deterministic systems, the residuals can be evaluated by comparing the norm of the residual signal against predetermined threshold values. In these norm-based approach, various norms can be applied for example L_2, peak and also root mean square (RMS) [218]. These norm-based approach not only require less computation, in addition they also allow for a systematic approach for computing threshold.

2.4.5 Energy-based Detection

Compared to observer-based detection, the use of system concepts such as passivity and/or energy in model-based detection is not very common. There are only a handful of work whereby the concept of energy is used in detection. In [221], the authors proposed a fault detection and isolation method for port-Hamiltonian systems to detect variations in the parameters of system components. The work in [222] proposed an energy balance scheme for fault detection for continuous-time passive systems. The author performed fault detection by checking when the energy balance is perturbed indicating the presence of faults. An energy balance fault detection approach was also applied for sensor fault detection in steel galvanizing process[223]. In this work the authors developed a FDI scheme to detect faults in single-input single-output closed-loop system without the requirement of an accurate state or dynamic system model. In [224], a passivity-based fault detection method was introduced based on evaluating the traditional passivity-based inequality. In this work, a fault is said to have occurred whenever the inequality is not satisfied.

2.4.6 Security of NCS

Over the past few years, given the prevalent attacks on NCS infrastructure, the security of NCS has received increased attention. With this increased attention, there has been numerous efforts towards securing NCS. NCS are typically designed to achieve certain operational goals such as closed loop stability, liveness or optimization of certain performance function. As described in [225], the main objective of the recent efforts towards securing NCS is to protect these operational goals as well as other non operational goals for e.g. privacy. Security goals can be generally described by; integrity which represents the trustworthiness of data or resources, availability which means the ability to access and use information on demand as specified; and confidentiality which is the ability to keep information secret or private from unauthorized users. Unlike other IT systems where securing against cyber attacks essentially involves encryption and protection of data, security of NCS is more complicated considering that malicious attacks in these systems can potentially influence the physical system's behavior. In order words, merely applying IT security measures in NCS, would not be enough to properly secure their correct operation against cyber attacks [226, 225].

The work in [227, 226, 225], discussed the fundamental vulnerabilities, threats and challenges facing NCS as a result of their complex and typically distributed architectures arising from the interconnection of IT and control systems. Prevalent malicious attacks on NCS can be categorized into denial-of-service (DoS) and integrity/deception attacks. The lack of the security goal of availability results in a DoS attack while the lack of the security goal of trustworthiness results in integrity/deception attacks These attacks are often considered intelligent being that they cannot be easily detected by simple bad data detection (BDD) schemes which are typically applied to detect outliers in measurement data based on a high fidelity model.

In [228], the authors presented a three-dimensional attack space linking different types of attacks both in the cyber and physical layers of NCS to the attackers' knowledge of the system as well as the disruption and disclosure resources available to the attackers. This detailed view provides an insight for both an attacker and defender on how an attack can be perpetuated as well as the potential impact of the attack on a system. As an illustrative example, the authors used a quadruple tank process as a testbed for staged cyber attacks in order to demonstrate different attack scenarios as captured by the attack space.

In [229], the authors consider DoS attacks whereby an attacker, with constrained resources, jams the communication channel. The formulation is proposed in a game theoretic framework. In this approach the attacker/jammer, for a limited number of time slots can jam the communication between the plant and the controller. A saddle-point equilibrium between the attacker and controller is shown to exist for the dynamic zero-sum game and in addition, for a specific instance problem, they proposed an optimal jamming policy which is proven to be of a threshold type. The authors in [230], consider NCS with safety constraints under DoS attacks. In their model, they considered the problem of finding an optimal feedback controller that minimizes a performance function in order for safety and power specifications to be satisfied. They also presented the effects of attack models on the solutions of the optimal control problem. In [231], the authors proposed two queuing models to simulate the stochastic process of packet delay jitter and loss under DoS attacks. Using these models, they quantitatively investigated the degradation of performance due to DoS attacks and proposed mitigating techniques based on packet filtering.

Integrity attacks such as deception has received great attention. These type of attacks typically involve the attacker accessing and modifying exchanged data in an effort to disrupt the nominal operation of the system. In [232], the authors analyzed the effects of false data injection attacks on control systems and provided necessary and sufficient conditions under which an attacker can destabilize a system while remaining undetected. The same authors presented a specific class of integrity or deception attacks called replay attacks in [233]. The replay attacks were demonstrated on state estimation in wireless networks and then a novel detection scheme was developed in order to handle such attacks. In [234], the authors consider false data injection attacks against state estimation in electric power grids. The authors show that by using the configuration of the power system, an attacker can launch successful attacks that introduce errors into certain state variables while bypassing existing BDD. They also consider realistic attacks whereby the attacker resources are constrained. In [235] the authors showed that the safety constraints of an automatic generation control (AGC) for power system under deception attack could be violated. The authors showed how to robustly interrupt the AGC signals and introduce an appropriate fake signal. In [236], the author described and characterized scenarios demonstrating stealthy attack policies whereby an attacker having detailed model knowledge and full access to all sensor and actuator channels is able to perform both disclosure and deception attacks. Then using a water irrigation example, the stealthy attack policies were illustrated.

The work in [237] describes a set of stealthy false-data injection attacks for omniscient attackers with full-state information compromising a subset of existing sensors and actuators. Stealthy deception attacks were also considered in networks distributively computing linear functions, where each node is modeled as a first-order system [238, 239]. From a system theoretic standpoint the stealthy deception attacks were characterized in terms of the number of compromised nodes and the network connectivity. The works in [240] also analyzes fundamental conditions that guarantee resilience in presence of adversaries. In [241], the authors considers the secure estimation and control of linear deterministic systems under malicious sensor attacks. The authors formulated the problem as a dynamic error correction problem with sparse vectors and showed that by using state feedback the resilience of a system can be increased.

In the security of NCS, the impact of an integrated security mechanism on the performance of the NCS is often neglected. In fact, there exists trade-offs between security and performance requirements due to the limited system resources and extra time delay imposed by the security additions. In [242], the basis effect of security on NCS performance is characterized. Additionally, the integrated security features are mapped to time delay in a system to show the trade-off between adding security and real-time operation for an NCS path tracking application. Also, in [243] a model for the trade-off is introduced and co-evolutionary algorithms are used to optimize this tradeoff.

From the aforementioned works, it is clear that in order to properly secure NCS, appropriate tools and techniques are required in order to understand and subsequently protect NCS infrastructure against malicious cyber attacks.

2.5 Applications

In this section, we review some representative applications. In particular, we briefly review two types of robotic systems: *robotic manipulators* and *wheeled mobile robots*. We also provide a very brief overview on *automotive systems*. These three applications embody the various challenges in designing NCS, from understanding the physical equations governing the dynamics of these systems as well as the emerging interactions due to the introduced communication networks and platforms, to the design and implementation of control techniques to achieve various objectives. Also, these applications present representative platforms to test various design approaches in order to validate research findings.

2.5.1 Robotic Systems

2.5.1.1 Robotic Manipulators

Robotic manipulators are highly nonlinear and coupled dynamic systems typically designed to move and maneuver objects within a given work environment using a number of actuated joints. A robotic manipulator's degree of freedom is equivalent to the number of joints (motors) and the type of joints it has. Robotic manipulators have been extensively applied in factory automation, space exploration, surgery, search and rescue missions, hazardous environment and other various military applications.

An n-degrees of freedom robotic manipulator can generally be described using the Euler-Lagrange equations of motion as [19]:

$$B(q)\ddot{q} + C(q,\dot{q})(\dot{q}) + F_v(\dot{q}) + G(q) = \tau; \tag{1}$$

where $q \in \mathbb{R}^n$ denotes the vector of generalized coordinates (rotational joint configurations), $B(q) \in \mathbb{R}^{n\times n}$ is the symmetric positive definite inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ is the matrix of centrifugal and Coriolis torques, $F_v(\dot{q}) \in \mathbb{R}^{n\times n}$ is the the diagonal matrix of the viscous friction coefficients, $G(q) \in \mathbb{R}^n$ is the gravitational torque vector. $\tau(t) \in \mathbb{R}^n$ is the input torque vector.

One fundamental objective that is necessary for the application of robotic manipulators in the various applications mentioned earlier is *trajectory tracking*. Trajectory tracking control of a robotic manipulator is the control of a robotic manipulator to follow a desired trajectory by the adjustment of the manipulator's joints. Trajectory tracking control of a robotic manipulator has attracted considerable interest over the past years. Numerous design techniques have been developed to address this problem. Traditional proportional-derivative (PD) controllers, have been shown to be simple and effective for set-point tracking [244]. However, for the case of trajectory tracking, PD controllers often require very large actuation to achieve precise control which is often not practical. In [245], various extensions of the PD control approach have been developed to improve the trajectory tracking ability of robotic manipulators.

In order to consider the impact of model uncertainties online parameter-estimation techniques in the form of adaptive control techniques have been sought for trajectory tracking control of robotic manipulators [246]. Robust controllers such as sliding mode, backstepping and passivity-based controllers have also been proposed towards addressing the trajectory tracking problem in the control of robotic manipulators. An exhaustive survey for these robust control approaches can be found in [247, 248]. The difficulty often involved in determining the dynamic model of robotic manipulators has resulted in the use of soft computing techniques in solving the trajectory tracking problem. These techniques include soft computing methods such as fuzzy-logic control, neural networks and genetic algorithms. The authors in [249, 250] provide a survey on the use of soft computing techniques in robot control which have been shown to be quite effective especially without a-priori knowledge of the system dynamics.

2.5.1.2 Wheeled Mobile Robots (WMRs)

Wheeled mobile robots (WMRs) represent another group of well-known robotic systems. Research involving wheeled mobile robots have been an active area of research over the past few years. The great interest in this area has been mainly fueled by the numerous practical applications that can be uniquely addressed by mobile robots due to their ability to work autonomously in large potentially unstructured and hazardous domains. A wheeled mobile robot is able to perceive its current conditions and that of its environment through an array of sensors. These information aid the WMR to autonomously navigate its environment with a set of motorized actuated wheels. WMRs have been employed for applications including: security, military operations, surveillance, mining operations, planetary exploration and even household cleaning. In these applications, WMRs operate in unknown environment where human intervention is expensive, monotonous, unreliable or impossible. Research in WMRs can in general be divided into several components namely, the modeling of the WMR, the planning and navigation strategies, the communication system, control tasks and localization techniques. We are particularly interested in the research efforts related to the control of WMR and the communication system to exchange information between the WMR's components as well as with other WMRs. The control of WMRs involves understanding the physical mechanics of the WMR platform, the model of the interaction between the WMR and it's environment as well as the impact of control algorithms on the WMR.

The control of WMRs to perform motion objectives is very important. The nonholonomic constraints associated with these systems have motivated research efforts towards techniques that effectively achieve the desired control objectives. Nonholonomic systems have motion limitations resulting from their kinematic model. Therefore, some directions of motion are not possible. There are many different types of WMRs. [251] provides structural properties of the kinematic and dynamics models of various types of wheeled robots. The authors then used the concepts of degree of mobility and degree of steerability of WMR in order to classify them into respective groups. A thorough analysis in regards to the classification is also provided.

The two most common types of WMRs are the unicycle-type WMRs and the car-like WMRs. The unicycle-type WMR is typically composed of two independently actuated wheels on a common axle whose direction is rigidly linked to the robot's chasis, and one or several passively orientable - or caster- wheels, which are not controlled. The car-like WMR is composed of a motorized wheeled axle at the rear of the chasis and one(or a pair of) orientable front steering wheel(s). The motion tasks of a WMR consists of three main generic control problems, path following, stabilization of trajectories and stabilization of fixed postures [252]. In path following, based on a predetermined velocity the problem involves driving on a road while trying to maintain the distance between the WMR chassis and the side of the road constant. An example application is the automatic wall following. In stabilization of trajectories, unlike path following, the velocity is not predetermined and the problem is typically formulated as the one of controlling the WMR to track a reference vehicle whose trajectory is given by $(x_r(t), y_r(t))$. The stabilization of fixed posture involves correct orientation of a WMR with respect to a desired reference posture.

There are numerous existing approaches for controlling WMRs to achieve motion tasks. In [253], the authors proposed a nonlinear feedback action in order to address the trajectory tracking problem. [254] provided alternative solutions to trajectory tracking using dynamic state feedback linearization. A recursive technique for trajectory tracking of nonholonomic systems in chained form was proposed in [255] through the extension of the backstepping control paradigm. For posture stabilization, both discontinuous and/or time-varying feedback controllers have been proposed. Smooth time-varying stabilization was pioneered by Samson in [256] with application also to the path following problem, while discontinuous (often time-varying) control have been proposed in various forms in [257, 258, 259]. Additionally, many other control techniques have been proposed in tackling the stabilization and path-tracking objectives such as model-based predictive controllers [260], Lyapunov-based controllers [261] and fuzzy neural networks [262].

WMRs have also been used extensively in various multi-agent cooperative control objectives such as flocking and platooning. The work in [46, 47, 263, 264] provides a review of multi-agent cooperative control objectives involving WMRs.

2.5.2 Automotive Systems

Fueled by advent of digital technology over the last few years, a typical modern automotive vehicle features the use of electronic systems to replace hydraulic and mechanical implementations of system functionalities. The electronic systems in an automotive system can be grouped into two main classes: the first category is the *hard-real-time control of mechanical parts* and the second category is information-entertainment (typically known as infotainment). The first category includes chassis control, automotive body(interior air conditioning, dashboard, power windows), powertrain(engine, transmission, and emission control systems) and active safety control systems. The second category includes information management, navigation, computing, external communication and entertainment. These modern automotive systems are equipped with up to 80 electronic control units (ECUs) exchanging more than 2500 signals over up to 5 different bus systems [265][266]. These ECUs control and monitor many of the aforementioned functionalities of a vehicle.

The development of control software implementing the aforementioned system functionalities has become one of the greatest challenges in the automotive domain due to the increasing complexity of automotive systems [267] and the increasing roles of control, computing and communications. Several works such as in [268, 269, 270], have shed light on the increasing roles of control, computing and communication as well as the benefits and challenges that they present. The development of control software involves understanding the dynamics of the vehicle with respect to the required system functionality, designing the controller functionality to achieve the design specification, and the testing, generation and deployment of the software components that implement the desired system functionality.

An exorbitant amount of research effort has been devoted towards understanding the dynamics of a vehicle in order to be able to design effective control techniques for various system functions. The work in [271] provides a thorough analysis and review of the various dynamics of a vehicle, especially the physical equations governing the main components of a vehicle including the tires, engine, transmission, brakes and suspension etc and how they interactively result in vehicle motion. The author also describes the performance of an automotive vehicle using basic foundation of engineering principles and analytical methods. The work in [272] provides a more recent and updated outlook on understanding the mechanics and dynamics of a vehicle. The author also provides a comprehensive coverage of automotive control systems for various vehicle system functionalities and describes various techniques in regards to the development of these control systems.

With the increasing complexity, high reliability and quality requirements, together with the pressure for reduced cost and short time to market, the development of automotive control software is typically based on the model-based approach together with platforms for testing. Model-based software development approaches as well as the design of test-bed for testing automotive control systems architecture is a very active research area and there's an increasing amount of research efforts in this area attempting to address various challenges highlighted in [273]. The authors in [274] provided a survey on the area of model based development and testing focusing mostly on the mainstream projects. Some research efforts in this area are devoted towards developing platforms for testing such as in [275], an automotive test-bed for electronic controller unit testing and verification. The authors in [276] presented a software-based implementation and verification scheme for a FlexRay based automotive network. In [277], the authors used a technique called Instrumentation Based Verification (IBV) to design automated tests in order to check whether models developed in Matlab/Simulink satisfy specified requirements. The complexity and tight coupling between the various layers of design in automotive systems pose great challenges to the model-based development process. In other to tackle some of these challenges, various efforts have been devoted to standardizing interfaces between the various layers of the automotive systems to enable easier integration in the development of these systems.

In the effort towards standardization of automotive system components in order to reduce the complexities involving in automotive control system development, initiatives such as AUTOSAR have been established. Automotive Open System Architecture (AUTOSAR) initiative is a consortium with the goal of an open standard for the automotive software architecture, through a component-based architecture that can support the reuse and scalability of future automotive software. Additionally, other standardization efforts are also being pursued for example in the case of in-car communication network frameworks such as FlexRay and TTEthernet, with the goal of guaranteeing highly reliable deterministic and fault-tolerant performance[278].

2.6 Comparison to this Dissertation

The presented work in this dissertation addresses specific problems in view of the four challenges highlighted in Section 1.2. In particular, we seek to develop tools and techniques for the modeling, design, analysis and evaluation of *dependable NCS*.

In regards to the challenge of modeling, design and analysis of NCS that are robust to network effects, most of the works in the literature are faced with cumbersome models and in some case oversimplified assumptions about the network due to the tight coupling of the NCS design layers. In contrast, the main idea for our model-based approach for the design and analysis of NCS is that by imposing passivity on the component dynamics we can ensure global system properties such as stability. Additionally, the design becomes insensitive to network effects, thus decoupling the controller design and implementation design layers with respect to network effects. This separation of concerns empowers the model based design process to be used for networked control systems without imposing additional constraints on the communication protocols.

Pertaining to the challenge of limited network resources, in most of the existing works, the stability of the system is often directly tied to the sampling scheme. In contrast, we introduce a set of passive components that allow variable sampling intervals in order to facilitate adaptive sampling. These components generalize the use of any adaptive sampling scheme that generates sampling intervals. Also due to the passivity of the components of the NCS, stability of the overall system is guaranteed irrespective of the sampling scheme. Hence, our approach provides the benefits of ensuring stability as well as the efficient use of network resources by the ability to incorporate an adaptive sampling scheme in the passivity framework.

Further, our approach towards addressing the challenge pertaining to the evaluation of NCS is different from other tools in multiple aspects. First, our integration of Matlab/Simulink and ns-2 for the simulation of NCS is based on the HLA framework. As described in Section 2.3.3.3, the HLA is a standard for simulation interoperability that allows independently developed simulations, each designed for a particular problem domain, to be combined into a larger and more complex simulation. By using this framework, we address the correctness and validity concerns typically posed by other tools for NCS. Secondly, our model-based approach provides a clear model of NCS architecture and the information exchange between the control and networking subsystems of a NCS through the tight integration of models in the respective subsystem. Such a design-time modeling environment that supports NCS integration model is not available with existing tools. As a result, in the existing tools, the interactions between the control and networking subsystems are described in an ad-hoc manner, resulting in error-prone experiment execution. Finally, the design-time efficiency and automatic code generation based on DSMLs is a strong feature of our tool.

Finally, our proposed attack detection scheme extends the energy-based detection schemes to NCS in the presence of attacks. We generalize the energy-based formulation and characterization in the presence of attacks for dissipative systems and in addition, we illustrate the characterization for various passive systems, a special class of dissipative systems. Compared to most existing detection schemes, the energy-based approach provides direct information about the stability guarantees of the overall system other than just detecting the presence of an attack. Also, rather than focusing on just the system output, our approach focuses more on the property of the system which provides more valuable information about the system state.

CHAPTER 3

BACKGROUND ON DISSIPATIVITY AND PASSIVITY

The underlying foundations of the work presented in this thesis is deeply rooted in the concept of passivity, a special case of dissipativity. Hence, it is important to provide some background on these concepts. First, we introduce the concept of dissipativity and passivity. Subsequently, we present the fundamental properties of passive systems. Next, we review the various interconnections of passive systems. Further, we introduce the notion of passivity indices and various techniques for rendering a certain class of non-passive systems passive. Finally, we introduce some passivity-related concepts such as wave variable transformation, which are fundamental to the design of NCS.

3.1 Dissipativity and Passivity

Dissipativity and its special case, passivity, are mathematical abstractions describing the intuitive physical concepts of power and energy conservation in the input-output behavior of a system [279, 280, 99]. The general notion of dissipativity originally emerged in the field of electrical circuit theory, where it is noted that a network consisting of only inductors, resistors and capacitors, does not generate energy [281]. The ability to study the behavior of a system in terms of energy functions has an extraordinary value as it gives a physical and intuitive interpretation of important problems such as system stability. This ability makes dissipativity and passivity very powerful and robust tools for system analysis and control design [282, 283, 106]. As such, dissipative systems, as well as passive systems, exhibit robustness to unmodeled disturbances and uncertainties including those resulting from implementation and network effects. These desirable features also make dissipativity and passivity well-suited for NCS applications.

3.2 Definitions of Passivity

Passivity can be mathematically defined using two main approaches (i) a state-space approach (ii) an input-output operator-based approach. These two approaches are described briefly in the following sections.

3.2.1 State-Space Approach

This approach is based on the energy-based behavior introduced by Willems [282, 284]. In this approach, a systematic framework is used to describe dissipativity and passivity in terms of inequalities involving the existence of a storage function (energy stored by the system) and a supply function (external received energy). The underlying relationship between dissipative and passive systems is essentially the choice of a particular supply function. In order to proceed with the formal definitions using this approach, we first provide the description of a dynamical system. Subsequently, we define the supply rate of a system and then the description of dissipative and passive systems.

3.2.1.1 Continuous-Time Systems

First, we provide the following definition of a continuous time dynamical system.

Definition 3.1. *[282] A dynamical system can be described by the following state space realization*

$$H_c : \begin{cases} \dot{x} = f(x, u) \\ y = h(x, u) \end{cases} \tag{2}$$

where $x \in X \subset \mathbb{R}^n$, $u \in U \subset \mathbb{R}^m$ and $y \in Y \subset \mathbb{R}^m$ are the state, input and output variables respectively and X, U and Y are the state, input and output spaces respectively. For the purpose of denotation, $x(t) = \phi(t, t_0, x_0, u)$ represents the state at time t reached from the initial state x_0 at t_0.

The definition of a supply function is provided as follows.

Definition 3.2. *[282] The supply rate $W(t) = W(u(t), y(t))$ is a real valued function defined on $U \times Y$, such that for any $u \in U$ and $x_0 \in X$ and $y(t) = h(\phi(t, t_0, x_0, u))$, W(t) satisfies*

$$\int_{t_0}^{t_1} |W(t)| dt < \infty \tag{3}$$

Based on the above concepts, we can now provide a definition of a dissipative system.

Definition 3.3. *[282] A dynamical system, H_c, is said to be dissipative with respect to the supply rate $W(t)$ if there exists a non-negative real function $V : X \mapsto \mathbb{R}^+$, called the storage function,* *such that for all $t_1 \geq t_0 \geq 0$, $x_0 \in X$ and $u \in U$,*

$$\int_{t_0}^{t_1} W(t)dt \geq V(x_1) - V(x_0) \tag{4}$$

This inequality is called the *dissipation inequality* [285]. There are several interesting candidates for the supply rate function but a particular candidate function which simplifies the extension of the supply rate representation to passivity, is the *QSR dissipativity supply rate*. The QSR dissipativity supply rate is described as follows:

Definition 3.4. *[285, 282] Given constant matrices $Q \in \mathbb{R}^{m\times m}$, $S \in \mathbb{R}^{m\times m}$, $R \in \mathbb{R}^{m\times m}$ with Q and R symmetric, a dynamical system, H_c, is said to be QSR dissipative if it is dissipative with respect to the supply rate, W(t), given as*

$$W(u, y) = y^T Qy + 2y^T Su + u^T Ru \tag{5}$$

From the QSR dissipativity definition in (3.4), we can now deduce special cases of QSR dissipative systems which appear as a result of specific choices of the Q, S and R parameters: If the system, H_c, is QSR-dissipative then it is

1. Passive *if Q=0, S=$\frac{1}{2}I$, $R = 0$*
2. Strictly input passive (SIP) *if $Q = 0$, $S = \frac{1}{2}I$, $R = -\delta I$*
3. Strictly output passive (SOP) *if $Q = -\epsilon I$, S=$\frac{1}{2}I$, $R = 0$*
4. Very strictly passive (VSP) *if Q=$-\epsilon I$, $S = 0$, R=$-\delta I$*
5. Finite-gain stable (FGS) *if Q=-I, S=0, R=$\gamma^2 I$*

where ϵ and δ are positive scalars and γ is an arbitrary constant.

3.2.1.2 Discrete-Time Systems

We provide the following definition of a discrete-time dynamical system.

Definition 3.5. *A discrete-time dynamical system can be described by the following state space realization*

$$H_d : \begin{cases} x_{k+1} = f(x(k), u(k)) \\ y(k) = h(x(k), u(k)) \end{cases} \tag{6}$$

where $x \in X \subset \mathbb{R}^n$, $u \in U \subset \mathbb{R}^m$ and $y \in Y \subset \mathbb{R}^m$ are the state, input and output variables respectively and X, U and Y are the state, input and output spaces respectively.

Definition 3.6. *[286][287] A system, $\mathcal{H}_d$, given by (6) is said to be dissipative with respect to the supply function $W(u(k), y(k))$ if there exists a positive definite function $V(x)$ or V_k, called storage function, satisfying $V(0) = 0$ such that $\forall x_0 \in X$, $\forall k \geq k_0$, and all $u \in \mathbb{R}^n$ and with, $V_k = \frac{1}{2}x_k^T P x_k$*

$$V_{k+1} - V_0 \leq \sum_{k=0}^{N-1} W(u(k), y(k)) \tag{7}$$

Definition 3.7. *[288][289] A dynamic system, $\mathcal{H}_d$, is said to be QSR-dissipative if it is dissipative with respect to the supply rate,W given as*

$$W(u, y) = y^T Q y + 2y^T S u + u^T R u \tag{8}$$

where Q, S, R are matrices of appropriate dimensions with Q and R symmetric.
Similar to its continuous counterpart in (5), by choosing different different values for Q, S and R special cases of dissipativeness can be derived [290].

3.2.2 Input-Output Operator-based Approach

Typically referred to as “input-output passive mapping”, this approach is purely based on an input-output representation of the system. In this case, the system is described like an operator that relates the inputs with the output with no regards to the internal system structure [99, 291].

3.2.2.1 Continuous-Time Systems

First, we describe the L_2-space as well as the extended L_{2e}-space of functions and then we provide the formal definitions for the main types of passivity. A signal $y\colon \mathbb{R}^+ \to \mathbb{R}^m$ is in the L_2-space of functions if it has finite energy; i.e.,

$$\int_0^\infty y^\mathsf{T}(t)y(t) < \infty.$$

L_2 includes only those functions that approach zero asymptotically. Due to this fact, we consider the extended L_2-space of functions, denoted L_{2e}. For a signal to be in L_{2e}, it must satisfy the condition that any finite truncation of the signal has finite energy. This implies that L_{2e} includes all signals without finite escape time. Next, we provide a definition of an inner product as well as its truncation.

Definition 3.8. *[282] The inner product is defined as follows:*

$$\langle u, y\rangle = \int_0^\infty u^\mathsf{T}(t)y(t)dt \tag{9}$$

and in the case of a truncation, $T \in \mathbb{R}^+$ which nullifies function values larger than T, the resulting inner product becomes

$$\langle u, y\rangle_T = \int_0^T u^\mathsf{T}(t)y(t)dt \tag{10}$$

Also

$$\langle y, y\rangle_T = \int_0^T y^\mathsf{T}(t)y(t)dt = \|(y)_T\|_2^2 \tag{11}$$

Using the above definitions, passivity is defined as follows:

Definition 3.9. *[99] Let $H_c\colon L_{2e}^m \to L_{2e}^m$. Then, for all $u \in L_{2e}^m$:*

1. *H_c is **passive** if there exist some constant β such that such that*

$$\langle H_c u, u\rangle_T = \langle y, u\rangle_T \geq -\beta, \quad \forall T \geq 0; \tag{12}$$

2. *H_c is **strictly output passive** if there exist constants β and $\epsilon \geq 0$ such that*

$$\langle H_c u, u\rangle_T = \langle y, u\rangle_T \geq \epsilon\|(y)_T\|_2^2 - \beta, \quad \forall T \geq 0; \tag{13}$$

3. *H_c is **strictly input passive** if there exist constants β and $\delta \geq 0$ such that*

$$\langle H_c u, u\rangle_T = \langle y, u\rangle_T \geq \delta\|(u)_T\|_2^2 - \beta, \quad \forall T \geq 0; \tag{14}$$

4. *H_c is **very strictly passive** if there exist constants β and ϵ, $\delta \geq 0$ such that*

$$\langle H_c u, u\rangle_T = \langle y, u\rangle_T \geq \delta\|(u)_T\|_2^2 + \epsilon\|(y)_T\|_2^{-}\beta, \quad \forall T \geq 0; \tag{15}$$

5. *H_c is **finite-gain stable** if there exist constants β and γ such that*

$$\langle H_c u, H_c u\rangle_T = \langle y, y\rangle_T \leq \gamma^2\|(u)_T\|_2^2 - \beta, \quad \forall T \geq 0; \tag{16}$$

3.2.2.2 Discrete-Time Systems

For the discrete time counterpart, we also describe the l_2 space as well as the extended l_{2e} space of functions and then we proceed to the passivity definitions. A signal $y\colon \mathbb{N}^+ \to \mathbb{N}^m$ is in the l_2-space of functions if it has finite energy; i.e.,

$$\sum_{k=0}^{\infty} y^{\mathsf{T}}(k)y(k) < \infty.$$

Similar to the continuous time counterpart, let the extended l_2-space of functions that map $\mathbb{N}^+$ to $\mathbb{N}^m$ be denoted as l_{2e}^m. These signals are mapped onto l_2^m by the N-truncation operator $(\cdot)_N\colon l_{2e}^m \to l_2^m$, which nullifies function values for indices larger than $N-1$. Further, for all $y, u \in l_{2e}^m$ define $\langle y, u\rangle_N \triangleq \sum_{k=0}^{N-1} y^{\mathsf{T}}(k)u(k)$ and $\|(y)_N\|_2^2 \triangleq \langle y, y\rangle_N$. The *mapping* H_d is a subset of $l_{2e}^m \times l_{2e}^m$ which identifies the set of possible outputs in l_{2e}^m for each input $u \in l_{2e}^m$.

Definition 3.10. *[292, 99, 279] Assuming all signals are assumed to be zero for all $k < 0$. Let $H_d \subset l_{2e}^m \times l_{2e}^m$. Then,*

1. *H_d is **passive** if there exists some constant $\beta \in \mathbb{R}$ (the bias) such that*

$$\langle y, u\rangle_N \geq -\beta, \forall N \in \mathbb{N}; \tag{17}$$

2. *H_d is* ***strictly output passive*** *if there exists some constants $\beta \in \mathbb{R}$ and $\epsilon > 0$ such that*

$$\langle y, u\rangle_N \geq \epsilon \|(y)_N\|_2^2 - \beta, \forall N \in \mathbb{N}. \tag{18}$$

3. *H_d is* ***strictly input passive*** *if there exists some constants $\beta \in \mathbb{R}$ and $\delta > 0$ such that*

$$\langle y, u\rangle_N \geq \delta \|(u)_N\|_2^2 - \beta, \forall N \in \mathbb{N}. \tag{19}$$

4. *H_d is* ***very strictly passive*** *if there exists some constants $\beta \in \mathbb{R}$ and $\epsilon, \delta > 0$ such that*

$$\langle y, u\rangle_N \geq \delta \|(u)_N\|_2^2 + \epsilon \|(y)_N\|_2^2 - \beta, \forall N \in \mathbb{N}. \tag{20}$$

5. *H_d is* ***finite-gain stable*** *if there exists some constants $\beta \in \mathbb{R}$ and γ such that*

$$\langle y, u\rangle_N \leq \gamma^2 \|(u)_N\|_2^2 - \beta, \forall N \in \mathbb{N}. \tag{21}$$

3.3 Passivity Properties

The vast applicability of passivity can be attributed to the highly desirable properties exhibited by passive systems which in turn simplifies system design and analysis. Passivity is well-suited for applications in linear or nonlinear, time-invariant or time varying, continuous or discrete time and distributed systems. Even in the case when the physical meanings of the storage function and supply rate are not explicit in describing passivity, abstract energy can be used in describing the storage function and supply rate for a system. The following highlights the fundamental properties of passive systems.

3.3.1 Stability

Generally, passivity is very useful in investigating stability of systems. Passivity of a system is typically related to the system's Lyapunov stability. Specifically, when a system is passive with respect to a positive definite storage function, the origin of the system is considered stable. Hence, in this case passivity is a sufficient condition for stability. On the other hand, if the associated storage function is only semi-definite, additional conditions on zero-state detectability is required to ensure stability of the system. The following is a characterization of the notion of zero-state detectability [293].

Definition 3.11. *A system is said to be zero-state detectable (ZSD) if for any x∈ X*

$$y(t) = h(\phi(t, t_0, x_0, 0), 0) = 0, \forall t \geq t_0 \geq 0 \implies \lim x(t) = \phi(t, t_0, x_0, 0) = 0 \qquad (22)$$

In a local sense, the system is said to be ZSD if there exists a neighborhood of X_0 of 0 such that the above equation holds for all x $\in X_0$.

Hence, passive systems with only semi-definite storage functions with the additional condition of zero-state detectability establishes the relationship between passivity and lyapunov stability.

The stability of passive systems can also be viewed from an input-output stability perspective, without looking at the internal stability approach such as the Lyapunov approach [294]. In this formulation, we can say that a system is stable if bounded input energy supplied to the system results in a bounded output energy.

3.3.2 Phase-related Properties

Passive systems exhibit desirable phase-related properties. For continuous-time passive linear systems, in the frequency domain, their real parts are positive over all frequencies which implies that they are are phase bounded. The phase shift for passive systems is always within $[-90°, 90°]$[295]. For example, continuous-time system passive systems are minimum-phase and also possess low relative degree, $r < 2$. In order to clearly understand these conditions we provide the definitions of zero dynamics, minimum-phase and relative degree as follows:

(a) **Zero-dynamics:**

The concept of zero-dynamics is very important in the characterization of passive systems. The definition of zero-dynamics is provided as follows:

Definition 3.12. *[280] The zero-dynamics of a system are the state dynamics associated with the output fixed at zero,*

$$\dot{x} = f(x, u); 0 = h(x, u) \qquad (23)$$

(b) **Minimum Phase:**

For continuous-time linear systems, the minimum-phase property implies that there are no zeros on the right-side of the complex plane. In the case of discrete-time systems, the minimum-phase property implies that there are no zeros outside the unit circle. Minimum-has systems are typically easy to control. The general definition of minimum-phase systems for both linear and nonlinear systems is provided as follows:

Definition 3.13. *[280] A system is minimum phase if its zero dynamics are Lyapunov stable in a neighborhood around the origin.*

On the other hand, non-minimum phase systems are more difficult to control and often characterized as systems with unstable zero dynamics.

(c) **Relative Degree:**

The property of relative degree for both linear and nonlinear systems can be defined as follows

Definition 3.14. *[280] The relative degree of a system is the number of times the output equation must be differentiated before the input variables appear.*

For LTI systems, the relative degree is characterized by the difference between the order of the denominator and that of the numerator. While continuous-time LTI passive systems have a relative degrees less than two, discrete-time passive systems have relative degree zero.

The properties of minimum-phase and relative degree of passive systems are important in characterizing passive systems as well as in the passification of non-passive systems, which will be discussed in the Section 3.5.

3.3.3 Kalman Yakubovich Popov (KYP) Property

The Kalman Yakubovich Popov (KYP) property is very important in the characterization of passivity. The KYP property provides a set of necessary and sufficient conditions for a system to meet in order to be passive. KYP property provides more information than just the stability of passive system. It defines the input-output property of the system by relating the inputs and outputs to the states via the storage function. For linear systems, the KYP property establishes the equivalence between the frequency domain characteristics and state space properties of passive and dissipative systems by means of the positive real transfer functions.

1. **Continuous-Time Systems**

The positive realness of a system emerged from the network theory and was described in frequency domain based on the fact that the time integral of energy input to a passive network must be positive. Positive real systems are defined as follows:

Definition 3.15. *A system is said to be positive real if for all u$\in$ U, $t_1 \geq t_0 \geq 0$*

$$\int_{t_0}^{t_1} y^{\mathsf{T}}(t)u(t) \geq 0 \tag{24}$$

whenever x(t_0)=0 with the integral considered along the system trajectories.

Before, presenting the KYP lemma we define the notion of a system being minimal.

Definition 3.16. *A continuous-time LTI system is minimal if it is both controllable and observable.*

The KYP lemma is then provided as follows

Theorem 3.17. *[280] Let the system, H, be a minimal realization where A is Hurwitz, (A,B) controllable and (A,C) observable. If there exists a real symmetric positive definite matrix P and real matrices W and L satisfying*

$$PA + A^T P = -L^T L \tag{25}$$

$$PB = C^T - LW \tag{26}$$

$$W^T W = D^T + D \tag{27}$$

regarded as the passivity conditions, then the transfer function $H(s) = C(sI - A)^{-1}B + D$ is positive real, that is it satisfies the following conditions

(a) all elements of H(s) are analytic for Re(s)>0,

(b) H(jw)+H^T(-jw)$\geq$ 0 $\forall$ w $\in \mathbb{R}$, for which s=jw is not a pole of any element of H(s), and

(c) the eigenvalues of A on the imaginary axis are simple and the corresponding residues

$$\lim(s - s_0)H(s) \tag{28}$$

(s_0 a pole of H(s)) are Hermitian and non-negative definite matrices.

Conversely if H(s) is positive real, then for any minimal realization of H(s), there exists P>0, W and L which satisfy the passivity conditions

The above definition can be equivalently represented in terms of linear matrix inequalities. From the above theorem, finding a $P = P^T \geq 0$ in Theorem 3.17 is equivalent to finding a P that is feasible for the following LMI constraints. Assuming a quadratic storage function, $V = \frac{1}{2}x^T Px$ Linear Matrix Inequality (LMI) constraints [296].

$$\begin{bmatrix} A^T P - PA - \hat{Q} & PB - \hat{S} \\ (PB - \hat{S})^T & -\hat{R} \end{bmatrix} \leqslant 0$$

$$\begin{aligned} where \quad & \hat{Q} = C^T QC, \quad \hat{S} = C^T S + C^T QD \\ & \hat{R} = D^T QD + (D^T S + S^T D) + R \\ & \exists \varepsilon > 0,\ Q = -\varepsilon I,\ R = 0,\ S = \frac{1}{2}I \end{aligned} \tag{29}$$

The KYP property can also be extended to nonlinear systems. Consider the following nonlinear system affine in the input

$$H : \begin{cases} \dot{x} = f(x(t)) + g(x(t))u(t) \\ y = h(x(t)) \end{cases} \tag{30}$$

where $x(t) \in (R)^n$. Also f, g and h are smooth functions of x.

Then the KYP lemma for the nonlinear system is defined as follows

Definition 3.18. *[297] Consider the nonlinear system (30). The following statements are equivalent.*

(1) There exists a $\mathcal{C}^1$ storage function $V(x) \geq 0$, $V(0) = 0$ and a function $S(x) \geq 0$ such that for all t ≥ 0:

$$V(x(t))V(x(0)) = \int_{t_0}^{t_1} y^T(s)u(s)ds - \int_{t_0}^{t_1} S(x(s))ds \tag{31}$$

The system is strictly passive for S(x) $>$ 0, passive for S(x) $\geq$ 0 and lossless for S(x)=0.

(2) There exists a $\mathcal{C}^1$ non-negative function V : X $\mapsto \mathbb{R}$ with $V(0) = 0$, such that

$$\begin{cases} L_f V(x) \leq S(x) \\ L_g V(x) = h^T(x) \end{cases} \tag{32}$$

where $L_g V(x) = \frac{\partial V(x)}{\partial x} g(x)$.

2. **Discrete-time Systems**

In discrete-time, the KYP property of a linear system is described by the following theorem:

Theorem 3.19. *[298] Let H(z) be a square matrix of real rational functions of z with no poles in $|z|$ >1 and simple poles only on $|z| \leq 1$, and let (A,B,C,D) be a minimal realization of H(z). If there exists a real symmetric positive definite matrix P and real matrices L and W such that*

$$A^T P A - P = -L^T L \tag{33}$$

$$A^T P B = C^T - LW \tag{34}$$

$$W^T W = (D + D^T) - B^T P B \tag{35}$$

then, the transfer function H(z) is discrete positive real, that is, satisfies the conditions

(a) H(z) has elements analytic in $|z|$ >1, and

(b) G(z)+G(z)$\geq$ 0 $\in |z|$ >1, the asterisk denotes complex conjugation.*

Conversely if H(z) is discrete positive real, then for any minimal realization of H(z), there exists P>0, W and L which satisfy the passivity conditions above. This lemma is also regarded as the *discrete positive real lemma.* A generalization of the discrete time positve real lemma or KYP lemma which includes the general notion of dissipative systems is called the *Generalized positive real lemma* and is defined as follows.

Lemma 3.20. *[290][**Generalized Positive Real Lemma**] Let $G(z)$ be a transfer function description and $M(z) = R + G^H(z)S + S^T G(z) + G^H(z)QG(z)$, with $G^H(z)$ denoting the hermitian transpose of G(z). Let $\mathcal{H}$ be a minimal realization of G(z). Then $\forall z$ s.t. $\|z\| \geq 1$, $M(z) \geq 0$ if and only if there exist a real symmetric positive definite matrix P and real matrices L and W such that*

$$A^T PA - P = C^T QC - L^T L \tag{36}$$

$$A^T PB = C^T QD + C^T S - LW \tag{37}$$

$$B^T PB = R + D^T S + S^T D + D^T QD + C^T S - W^T W \tag{38}$$

The above lemma can also be equivalently represented in the form of LMIs. Assuming a quadratic storage function $V = \frac{1}{2}x^T Px$

$$\begin{bmatrix} A^T PA - P - \hat{Q} & A^T PB - \hat{S} \\ (B^T PA - \hat{S})^T & -\hat{R} \end{bmatrix} \leqslant 0$$

$$\begin{aligned} where \quad \hat{Q} = C^T QC, \quad \hat{S} &= C^T S + C^T QD \\ \hat{R} &= D^T QD + (D^T S + S^T D) + R \end{aligned} \tag{39}$$

3.4 Interconnection of Passive Systems

The compositionality of passive systems refers to the ability of constructing large-scale systems that preserve passivity through the interconnection of other passive systems by following simple interconnection rules [299, 280, 293]. These passivity-preserving interconnection rules include the parallel interconnection, feedback interconnection and symmetric input-output transformation of passive systems and are illustrated in following:

3.4.1 Parallel Interconnection

The interconnection of two passive systems, A and B in a parallel configuration as shown in Figure 5, results in the system, H, with input $u = u_1 = u_2$ and output $y = y_1 + y_2$, that is equally passive from u to y [300].

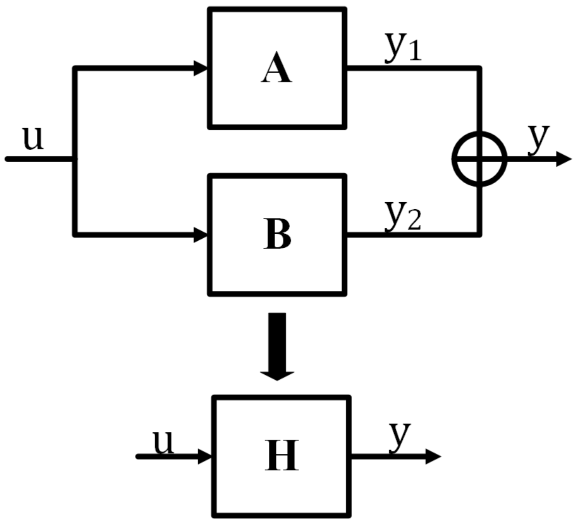


Figure 5: Parallel Interconnection of Passive Systems.

3.4.2 Feedback Interconnection

The feedback interconnection of passive is regarded as the most important passivity result and has been used extensively in numerous applications [293]. The feedback interconnection of two passive systems, A and B as shown in Figure 6, results in a system, H, with input and output, $u = u_1 + y_2$ and $y = y_1$ respectively, that is equally passive from u to y.

3.4.3 Symmetric Input-Output Transformation

A system composition where by the inputs of a passive system is pre-multiplied by a matrix and the output of the systems is post-multiplied by the transpose of the same matrix as shown in Figure 7, also preserves passivity [279]. Hence, the resulting system, H is passive from u to y.

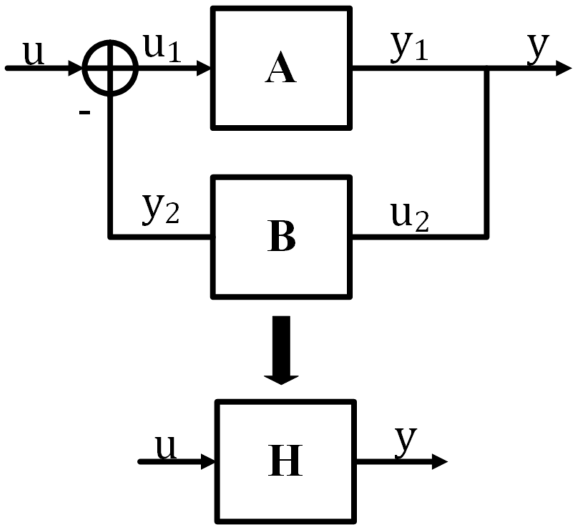


Figure 6: Feedback Interconnection of Passive Systems.

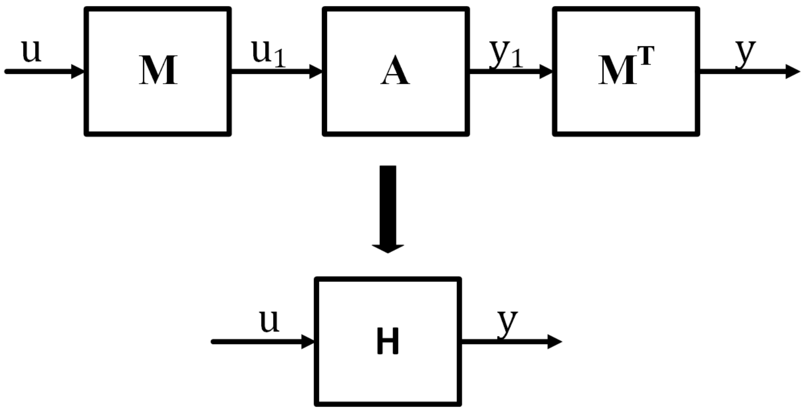


Figure 7: Pre- and Post- Multiplication of a Passive System.

These compositional rules form the basis for most of passivity-based control design techniques existing in the literature.

3.5 Passivity Indices and Passification

3.5.1 Passivity Indices

The concept of *passivity index*, which was first introduced by [301], quantifies the degree of passivity and is used to measure how far a system is from being passive. Passivity indices can be defined in terms of an excess or shortage of passivity. There are two main passivity indices that are used to measure the level of passivity of a given system, the *input feedforward passivity* (IFP) and *output feedback passivity* (OFP) [295] as shown in Figures 8 and 9 respectively. These passivity indices are defined such that a positive value for an index corresponds to an excess of that form of passivity while a negative value for an index indicates a shortage. Systems that are passive have positive or zero values for both indices.

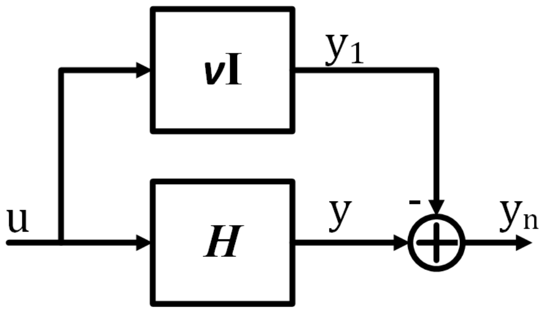


Figure 8: Input Feedforward Passivity

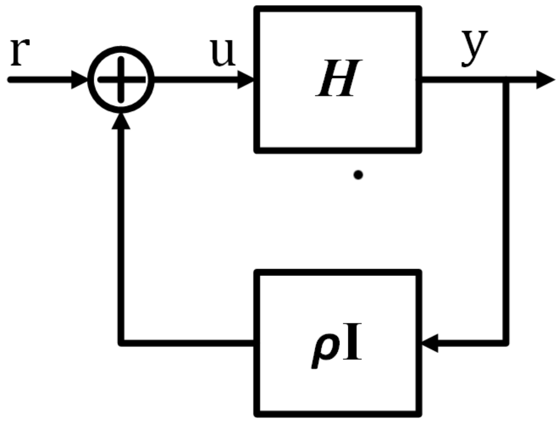


Figure 9: Output Feedback Passivity

3.5.1.1 Input Feedforward Passivity Index

The input feedforward passivity index (IFP) measures the extent of the minimum phase property of a system. It represents the largest gain that can be put in a negative parallel connection with a system such that the interconnected system is passive.

Definition 3.21. *[300] A continuous-time system, $H_c : u \mapsto y$, has input feedforward passivity*

(IFP) index, ν, if the following inequality holds $\forall t$

$$\int_0^T u^T y dt \geq S(x(T)) - S(x(0)) + \nu \int_0^T u^T u dt \tag{40}$$

Definition 3.22. *[300] A discrete-time system, $H_d : u \mapsto y$, has input feedforward passivity (IFP) index, ν, if the following inequality holds $\forall t$*

$$\sum_{i=0}^{N} u^T y \geq S(x(i+1)) - S(x_0) + \nu \sum_{i=0}^{N} u^T u \tag{41}$$

3.5.1.2 Output Feedback Passivity Index

The output feedback passivity (OFP) index is a measure of the level of stability of a system. The physical significance of this index is that it is the largest gain that can be placed in positive feedback with a system such that the interconnected system is passive. A positive value for the OFP is also a measure of the L_2 gain of a system which is computed in terms of the index value as $\gamma = \frac{1}{\rho}$ [99].

Definition 3.23. *[300] A continuous-time system, $H_c : u \mapsto y$, has output feedback passivity (OFP) index ρ if the following inequality holds $\forall t$*

$$\int_0^T u^T y dt \geq S(x(T)) - S(x(0)) + \rho \int_0^T y^T y dt \tag{42}$$

Definition 3.24. *[300] A discrete-time system, $H_d : u \mapsto y$ has output feedback passivity (OFP) index ρ if the following inequality holds $\forall t$*

$$\sum_{i=0}^{N} u^T y \geq S(x(i+1)) - S(x(0)) + \rho \sum_{i=0}^{N} y^T y \tag{43}$$

In other to characterize the passivity level of a system, both indices are necessary. Based on the two indices, a system's index can be denoted as (ρ, ν). The IFP and OFP indices are independent of each other and hence knowing one of the indices does not provide any information about the other except that the other index must exist. From the above definitions of passivity indices, it can be seen that a positive value of ρ implies *strictly output passive* as described in Section 3.2, while a positive value of ν implies *strictly input passive*. Additionally, these indices play an important role in rendering non-passive systems passive as will be illustrated in Section 3.5.2.

3.5.1.3 Computing Passivity Indices for Linear Systems

The passivity indices for linear system can be explicitly calculated based on the well known KYP lemma.

(i) **Continuous-time Systems**

For a continuous-time linear system, H_c, the two indices, if they exist, can be computed as follows [295]:

(a) IFP Index: For a stable linear system $H_c(s)$, the IFP index is given by the following frequency-dependent function

$$\nu(H_c(s), w) = \frac{1}{2}\lambda_{min}[H_c(jw) + H_c(jw)^*] \tag{44}$$

where λ_{min} represents the minimum eigenvalue.

(b) OFP Index: For a minimum phase linear system $H_c(s)$, the OFP index is given by the following frequency-dependent function

$$\rho(H_c(s), w) = \frac{1}{2}\lambda_{min}[H_c^{-1}(jw) + H_c^{-1}(jw)^*] \tag{45}$$

where λ_{min} represents the minimum eigenvalue.

(ii) **Discrete-time Systems**

For a discrete-time linear system, H_d, the two indices, if they exist, can be computed as follows [302]:

(a) IFP Index: For a stable linear system, $H_d(z)$, the IFP index is given by the following frequency-dependent function

$$\nu(H_d(z), \theta) = \frac{1}{2}\lambda_{min}[H_d(e^{j\theta}) + H_d(e^{j\theta})^*] \tag{46}$$

where λ_{min} represents the minimum eigenvalue.

(b) OFP Index: For a minimum phase linear system $H_d(z)$, the OFP index is given by the following frequency-dependent function

$$\rho(H_d(z), \theta) = \frac{1}{2}\lambda_{min}[H_d^{-1}(e^{j\theta}) + H_d^{-1}(e^{j\theta})^*] \tag{47}$$

where λ_{min} represents the minimum eigenvalue and $0 \leq \theta \leq \pi$.

3.5.2 Passification

In most cases, passivity-based techniques cannot be directly applied to systems which are not naturally or inherently passive. Under certain conditions, some of these non-passive systems can be rendered passive by compensating for their shortage of passivity. The approach of rendering a non-passive system passive is termed *passification*. Due to the benefits of passivity-based control, there are numerous efforts related to techniques in rendering systems passive. Based on the passivity indices described in Section 3.5.1, loop transformations, can be used for passification. A system that lacks OFP is unstable and can be made passive by using a negative feedback if the system is minimum phase and has low relative degree. Additionally, a system that lacks IFP is non-minimum phase and can be made passive with positive feedforward if the system is stable. For certain systems, such as a system that is unstable and non-minimum phase a combination of the loop transformation of feedback and feedforward gains cannot render the system passive and for this case the indices do not exist for this system[300].

In [303], the authors rendered a non-passive system passive by applying a state-feedback function as a passifying input to the system. [304, 305] describe the use of output feedback in order to render non-passive system passive. In [306, 307], the authors presented four passification techniques for linear systems by using four different compensators: feedback, series, hybrid and feedfoward compensators depending on the specific properties of the non-passive system.

3.6 Wave Variable Transformation

A very important tool that is typically associated with passivity, in particular during the exchange of information between systems is the wave variable formalism. The wave variable transformation originated from the scattering transformation in [23]. The scattering transformation defines a signal representation which is invariant to translations and potentially to other groups of transformations such as rotations or scaling. The fundamental idea behind the wave variable formalism is to transform two variables of the same dimension, often the input and output variables of a system, so that the resulting variables are linear combinations of the original variables. In wave variable transformation as shown in Figure 10, variables are transformed into transmitted and reflected waves, and thus are called *wave variables*. The equations, (48) and (49), denotes the relationship between the wave variables (u,v) and the actual system variables (e,f). Wave variable transformation has an underlying powerful intuition when the input and output variables of a system are power variables represented as effort and flow variables such as force and velocity. In this case, the physical unit of the square of each wave variable is Watts, and the direction of transmission of each wave variable represents *power flow* [308].

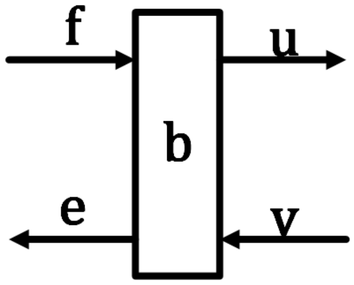


Figure 10: Wave Variable Transformation.

$$u = \frac{1}{\sqrt{2b}}(f + be) \tag{48}$$

$$v = \frac{1}{\sqrt{2b}}(f - be) \tag{49}$$

Then, the difference of the transmitted and reflected waves is the *net power flow*. In the case where the physical units do not admit this interpretation of power flow, the notion of *abstract power flow* allows the intuition to be generalized. Wave variables are very beneficial because the energy of transmitted waves are conserved when delayed by a constant value and provide robustness to delays. This desirable property makes wave variables well-suited for NCS applications, where undesirable network effects such as delays and packet loss adversely affect the overall system behavior.

CHAPTER 4

MODEL-BASED DESIGN OF PASSIVITY-BASED NCS

4.1 Introduction

Model-driven development has been found to be very beneficial in the systematic design and analysis of complex systems. Model-based design for embedded control systems involves creating models and checking correctness at different stages in the development process [160]. The typical design flow progresses along precisely defined abstraction layers, typically starting with control design followed by system-level design for the specification of platform details, code organization, and deployment details, and the final stage of integration and testing on the deployed system. The complexity and heterogeneity due to the introduction of NCS architectures make it extremely challenging to use these existing model-based design approaches to effectively construct CPS. The tight coupling between design concerns create a number of challenges. Ensuring controller stability and performance for physical systems in the presence of network uncertainties (e.g. time delay, packet loss) couples the control and system-level design layers making it difficult to guarantee system properties. In addition, downstream code modifications during testing and debugging invalidate results from earlier design-time analysis and any component change often results in “restarting” the design process.

In an effort towards addressing these complexity and heterogeneity challenges in the design of NCS, we propose an automated model-based approach based on the system theoretic concept of passivity. The primary idea is that by imposing passivity constraints on the control design, which in effect decouples the control design from network uncertainties such as time delays and packet loss, we provide a simplification strategy that limits the complexity of interactions and empowers the use of model-based approaches. Our main goal is to used the principle of decoupling to demonstrate a model-based compositional framework for NCS. Using Model-Integrated Computing [160], we developed a domain specific modeling language (DSML) called the Passive Networked Control Systems (PaNeCS). PaNeCS raises the level of abstraction of NCS design and allows automated analysis, code generation, system configuration, deployment, and testing. More importantly, it facilitates effective engineering processes and methods for designing, building, and analyzing NCS by utilizing the compositionality across the design views stemming from the underlying passivity principles. The underlying passivity principles provides robustness to network delays and packet loss. Also, PaNeCS facilitates the rapid prototyping of networked control systems and enables testing, through running different experiments under various network conditions, simply by adjusting parameters and generating appropriate software for each configuration.

The main contribution of PaNeCS is that we developed a DSML for the compositional design of passivity-based NCS that are robust to network uncertainties such as time delays and packet loss. We integrated a component passivity analysis tool, which together with the encoded compositional rules in the language ensures a "correct-by-construction" NCS design. We also integrated model interpreters for transforming designed PaNeCS model into platform-specific models for simulations using Matlab/Simulink/TrueTime. Also, we developed a code generator for generating executables for implementing experiments for Euler-Lagrangian systems, specifically, networked multi-robot systems. Finally, we provide simulation and experimental results demonstrating the effectiveness of the approach and the robustness of the passivity-based approach.

The rest of the chapter is organized as follows: We briefly discuss a passive networked control architecture in Section 4.2. Section 4.3 presents an overview of PaNeCS. Section 4.4 presents the modeling language. Section 4.5 discusses an integrated analysis tool for automatically checking passivity of modeled components. Section 4.6 presents the model interpreters for the automated code generation of models for simulations as well as for experiments. Section 4.7 presents a simulation case study on the control of two linear plants over a wireless network. Section 4.8 presents an experimental case study on the control of a networked multi-robot system. Finally, a summary of the chapter is provided in Section 4.9.

4.2 Passivity-Based Control of Networked Control Systems

We briefly discuss the passivity based control architecture for multiple plants controlled by a single controller or multiple controllers via a network [309]. Figure 11 depicts a simple networked control system with only one plant shown. The wave transform block, denoted **b**, represents a transformation between signals and wave variables. In Figure 11, $u_{pk}(i)$ (where k=2,...,n), can be thought of as sensor output data in wave variable form from each plant, where $n-1$ is the total number of plants in the network. Likewise, $v_{cj}(i)$ (where j=1) can be thought of as a command output in wave variable form from the controller.

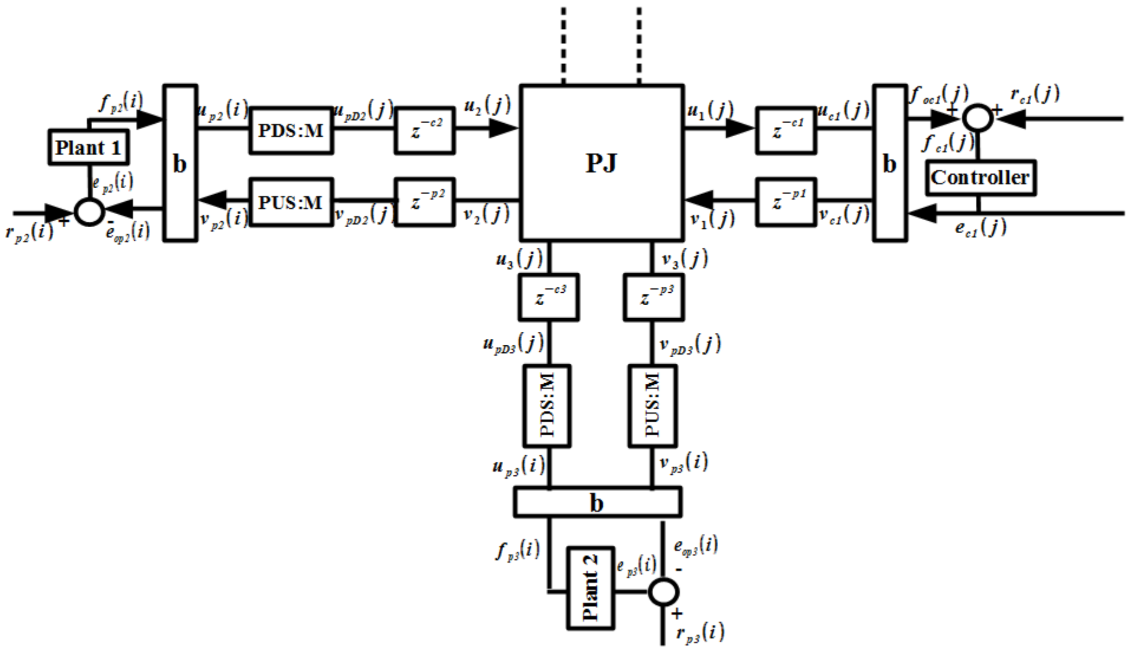


Figure 11: Networked Control Architecture

The power junction, denoted PJ in Figure 11, is used to interconnect wave variables from multiple controllers and multiple plants in parallel such that the total input power is always greater than or equal to the total output power. This provides a formal way to construct a NCS design. A power junction makes it possible for a single controller to control multiple plants over a network and guarantee that the overall system remains stable. In Figure 11, the power junction has waves entering and leaving as indicated by the arrows. The blocks, z^{-ck} and z^{-pk} (k=1,2), represent network delays incurred by the wave variables. Waves entering the power junction from the controller are network-delayed versions of waves leaving the controller, as indicated by the time delay block. Waves entering the controller are delayed versions of waves leaving the power junction. The other waves in the diagram are similarly delayed.

Due to bandwidth constraints, the controller typically runs at a slower rate than the sensors and actuators of the plants. In order to preserve passivity in the multi-rate digital control network we use the passive upsampler PUS:M and passive downsampler PDS:M pair to handle the data rate transitions. The PUS:M and PDS:M as shown in Figure 11 provide the upsampled and downsampled versions of their respective wave variable inputs while preserving passivity. The block parameter

M is the sampling ratio – the data rate of the faster side of the connection divided by the data rate on the slow side. Based on the architecture described in Figure 11, we can now describe our the modeling language, PaNeCS.

In [309, 6], the presented analytical results for the architecture in Figure 11 ensures passivity of the power junction, and if the the plants and controllers are passive, the overall NCS is guaranteed to be passive. Also, if the plants and controllers are strictly-output passive, the overall network is l_2-stable, a stronger robustness property than just being passive.

4.3 Overview of PaNeCS

The passivity-based modeling language, PaNeCS, is developed using the Generic Modeling Environment (GME), from the Model Integrated Computing (MIC) tool suite [310]. GME provides a metamodeling environment similar to UML. The class stereotypes are briefly defined as follows: Models are entities which may contain other objects while Atoms are indivisible entities which cannot contain other objects; Connections are association classes used to describe the relationship between two entities and they represent a line that connects two entities of a model. Connectors signified by "." specify a visualization for a connection in the model. Each of the entities associated with the connector have well defined roles (src and dst) with respect to the connector. These roles define the direction of the connection between the entities.

PaNeCS encodes passivity constraints into the language's structural semantics in order to achieve passive compositional design of NCS. Figure 12 shows the design flow in PaNeCS. PaNeCS supports the modeling of NCS composed of both linear and nonlinear dynamical plant systems. The integrated analysis tool in PaNeCS, denoted as **Passivity Analysis** in Figure 12 enables the verification of passivity for linear plants and controller specified as model components in PaNeCS. After modeling NCS in PaNeCS and verifying that the specified components satisfy passivity, simulation code for simulation of the NCS can be automatically generated for evaluation as denoted by the block **PaNeCS to Simulink/TT** in Figure 12. Also, as depicted by the block, **PaNeCS to Executables**, in Figure 12, executables for experimental evaluation of modeled NCS can be automatically generated. In the following sections, based on the components of the design flow as shown in Figure 12, we will describe the modeling, analysis and code generation in PaNeCS.

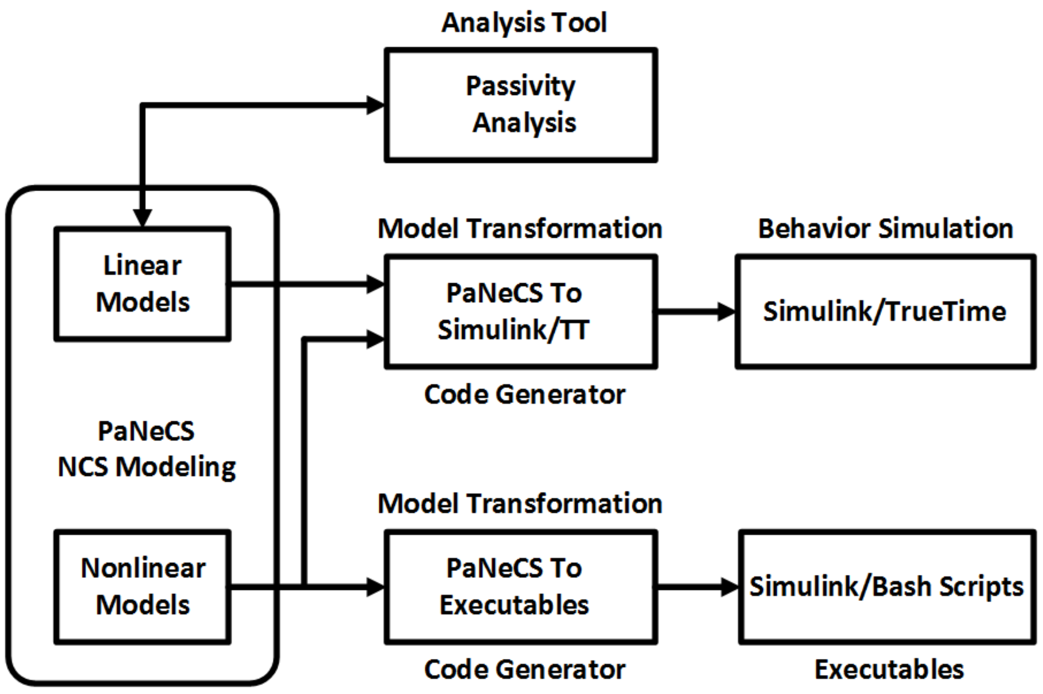


Figure 12: PaNeCS Design Flow

4.4 Modeling in PaNeCs

4.4.1 Components

In PaNeCS, the language top level consists of eight main components: the *PlantSystem*, *PhysicalPlant*, *ControllerSystem*, *PowerJunction*, *PhysicalReference*, *ReferenceSystem*, *Processor*, and *Network*.

(i) Plant System: PaNeCS can model two types of dynamical plant systems, linear systems and non-linear systems

(a) Linear Systems: In modeling a linear plant, *PlantSystem* represents all the sub-components on the plant side of the network. These components include *Plant*, *BilinearTransformP*, *PassiveUpSampler*, *PassiveDownSampler*, *Send* and *Receive*. *Plant* represents the system to be controlled. The *Plant* model can be any passive discrete linear time-invariant (LTI) system, $\mathcal{H}$,

$$\mathcal{H} : \begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k + Du_k \end{cases} \tag{50}$$

where $x_k \in \mathcal{X}$ represents the state variables, $u_k \in \mathcal{U}$ represents the control inputs to the plant and $y_k \in \mathcal{Y}$ represents the plant outputs obtained by sensors at sampling instant $k \in \mathbb{Z}$.

The *Plant* dynamics are parametrized by matrix attributes A, B, C, D, and a scalar *SamplingTime*, and are specified using any valid Matlab expression evaluating to the proper dimensions. *BilinearTransformP*, parametrized by impedance b, represents a model for wave scattering, which transforms the wave variables received from the power junction into control input to the plant and transforms the plant output signal into wave variables that are transmitted over the network. *PassiveUpSampler and PassiveDownSampler* pair represent the PUS:M and PDS:M pair discussed in Section 4.2. *Send* and *Receive* pair are used to represent the transmission of data over the network. Figure 13 shows a part of the PaNeCS metamodel that describes the plant subsystem.

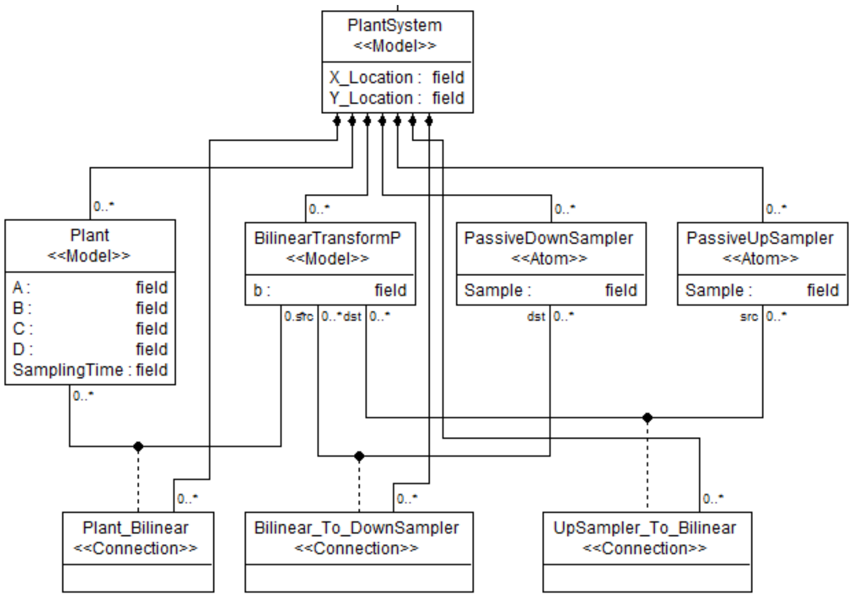
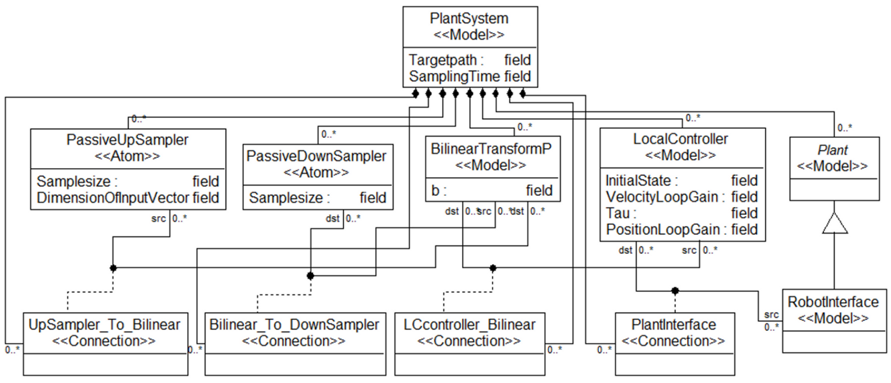


Figure 13: PlantSystem Sub-Language (Linear)

(b) Non-linear Systems: Figure 14 shows the parts of the metamodel that describe the *PlantSys-*

tem when modeling non-linear systems such as a robotic manipulator. *RobotInterface* depicts the software interface for connecting to the *PhysicalPlant*(an actual robotic system). *Local Controller* represents the software component for implementing the local control commands that are sent to the robot and also for processing the sensor data received from the robot. The *Local Controller* is parametrized by the initial state of the robot, q_0, the position loop gain, K_x, the velocity loop gain, K_v and the time constant, τ. These attributes are used for configuring the passive controller for the robot. Figure 15 shows the language of the *Ports* of the *PlantSystem*. The *Send* and *Receive* blocks indicate the ports for sending and receiving wave variables from the power junction respectively. The two blocks are both parametrized by the port number parameter denoted as *PortNo*. Additionally, the *Receive* block has a parameter, *VectorLength*, which specifies the length of the vector of the data that is received.



PlantSystem <<Model>>
Targetpath : field
SamplingTime field
PassiveUpSampler <<Atom>>
Samplesize : field
DimensionOfInputVector field
PassiveDownSampler <<Atom>>
Samplesize : field
BilinearTransformP <<Model>>
b : field
LocalController <<Model>>
InitialState : field
VelocityLoopGain : field
Tau : field
PositionLoopGain : field
Plant <<Model>>
RobotInterface <<Model>>
UpSampler_To_Bilinear <<Connection>>
Bilinear_To_DownSampler <<Connection>>
LCcontroller_Bilinear <<Connection>>
PlantInterface <<Connection>>
src
dst
0..*

Figure 14: PlantSystem Sub-Language (Non-Linear)

(ii) Controller System: The *ControllerSystem* component models all the sub-components on the controller side of the network. For controlling a linear plant, these include *DigitalController*, *BilinearTransformC*, *ZeroOrderHold*, *ReferenceInput*, *Send* and *Receive*. *DigitalController* is a model representing the algorithm for controlling the networked plants. Similar to the linear model of the plant in *PlantSystem*, the *DigitalController* is modeled as an LTI system. Therefore, the *DigitalController* parameters have similar attributes to the *Plant* for the LTI system case. *BilinearTransformC* is similar to the *BilinearTransformP* described in the *PlantSystem*. *ZeroOrderHold* represents a component that holds its input for the time period specified in

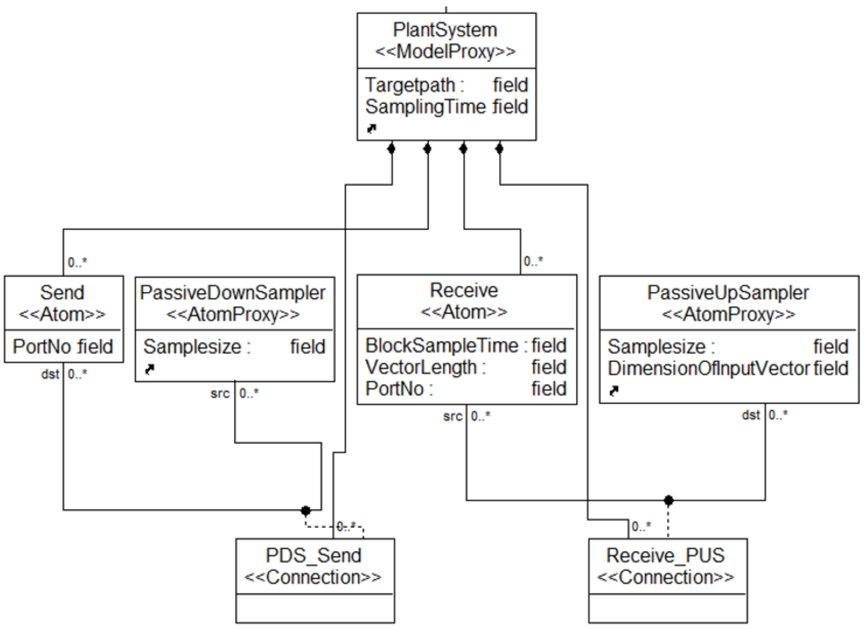


Figure 15: PlantSystem Ports Sub-Language

the sampling time attribute. *ReferenceInput* represents the desired signal to be tracked by the plants. Similar to the *PlantSystem*, *Send* and *Receive* pair are used to represent the transmission of data over the network.

For the nonlinear plant models, Figure 17 shows the parts of the metamodel which describe the *ControllerSystem*. In this case, the *Digital Controller* is parametrized by the control gain parameter, K and the control design frequency, w, the cutoff frequency, w_n and the damping coefficient, ζ for the filter used in smoothing out the reference trajectory. Additionally, in the *ControllerSystem*, a *Receive* block also indicates a port for receiving reference signals.

(iii) Power Junction: Figure 18 denotes the *PowerJunction* component of PaNeCS. The *PowerJunction* models components for implementing the interconnections of plants and controllers. The *PowerJunction* can two types of entities representing the two possible interconnection to the *PowerJunction*, the PowerInputPowerOutput and the PowerOutputPowerInput. The PowerInputPowerOutput entity models the software component for interconnecting plants and the power junction. Through this power port entity, the *PowerJunction* sends calculated wave transformed control signals to the *PlantSystem* and also receives wave transformed sensor signals from the *PlantSystem*. The PowerOutputPowerInput entity models the software com-

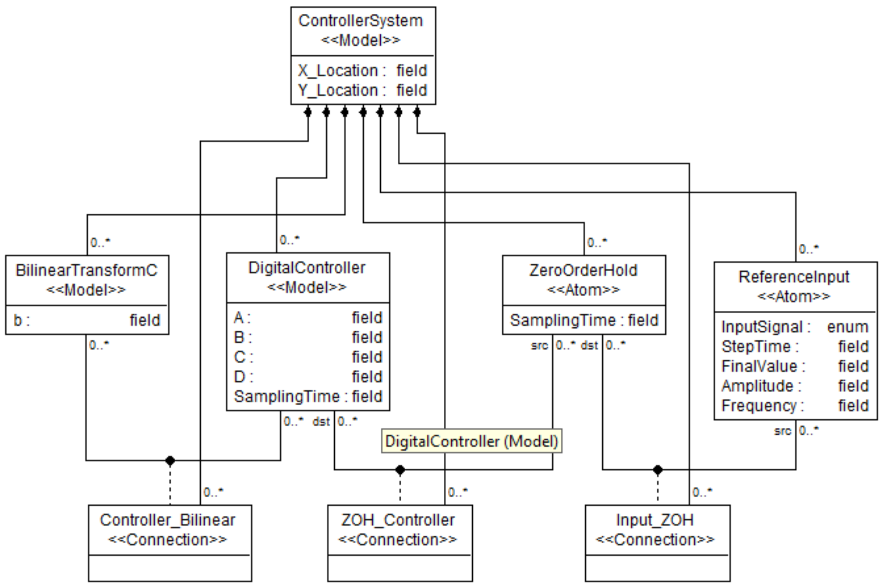


Figure 16: ControllerSystem Sub-Language (For Linear Plant)

ponents for interconnecting controllers and the power junction. Through this power port, the *PowerJunction* sends the averaged wave transformed sensor signal to the *ControllerSystem* and receives the calculated wave transformed control signal from the *ControllerSystem*. The PowerOutputPowerInput and PowerOutputPowerInput entities are both parametrized by send and receive port numbers denoted as SndPortNo and RcvPortNo. In running actual experiments, these port number attributes are used to specify the ports for sending and receiving information over an actual network.

(iv) *Reference*: Figure 19 denotes part of the language which describes the *ReferenceSystem*. The entity HapticInterface models the software interface for connecting to the *PhysicalReference*(which represents the Haptic Paddle) that can be used to generate the desired reference trajectory. The inverse kinematics entity models the software component for computing the inverse kinematics of the trajectory received from the haptic device. This component translates the task space coordinates of the haptic device defined by its *x*-*y*-*z* coordinates to the required joint angles for a modeled robotic arm [311]. The SendReference entity models the ports for sending the reference signal to a controller. This entity is parametrized by the port number denoted as SndPortNo.

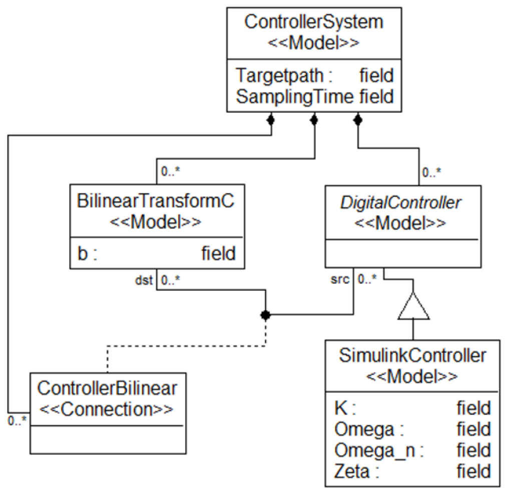


Figure 17: ControllerSystem Sub-Language (For Non-linear Plant)

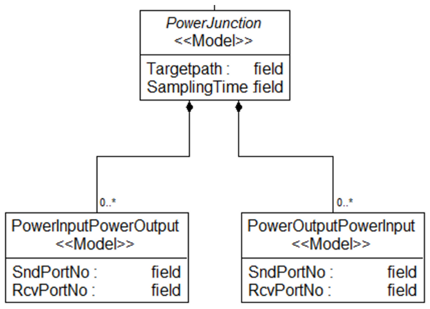


Figure 18: PowerJunction Sub-Language

(v) *Network*: The Network entity models the network used for the control system. This entity provides modifiable parameters for network configuration. The parameter NetworkType provides an option for choosing either to use wired or wireless network. The parameter DataRate sets the throughput for simulating network activity. DisturbancePacketSize configures the size of simulated disturbance attack packets on the network which facilitates the introduction of delays. This provides a way for simulating the NCS under non-optimal conditions. DisturbancePeriod configures the frequency of disturbance attacks on the network.

(vi) Processor: The *Processor* component models the computer on which the computations and software components are executed. It is parametrized by the IP address of the computer denoted as *IPAddress*.

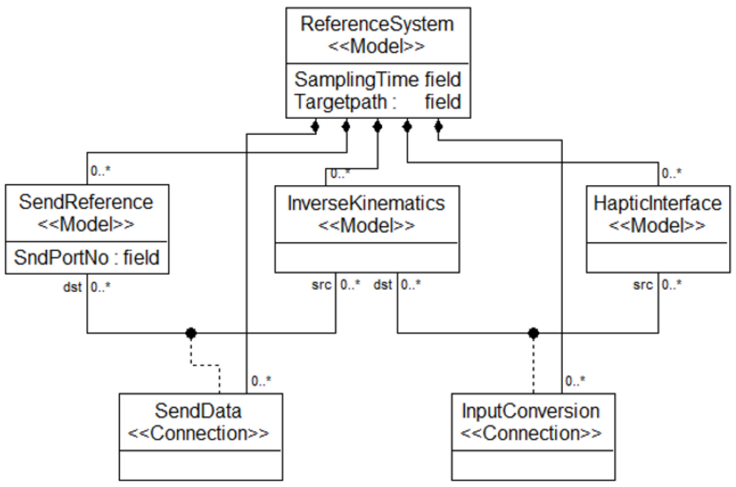


Figure 19: ReferenceSystem Sub-Language

4.4.2 Aspects

PaNeCS has three main design views or aspects: the *ControlDesign Aspect*, *Platform Aspect* and *ProcessorAssignment Aspect*.

(i) Control Design Aspect: The *ControlDesign Aspect* visualizes the controller modeling layer. This includes the plants, controller, and power junction, as well as their interconnections – indicating the flow of control and sensor signals.

(ii) Platform Aspect: The *Platform Aspect* visualizes the physical platform components of the NCS. This view includes plants, controllers, and the network as well as their interconnections – indicating the flow of data packets over the network. Though the plants and controller appear in both aspects, in the Platform aspect they represent physical devices rather than control design concepts.

(iii) Processor Assignment Aspect: The *ProcessorAssignment Aspect* depicts the mapping of the software components to processors on which the computations and implementations are to be performed. The entities in this view include the *Processors*, *PlantSystem*, *ControllerSystem*, *PowerJunction* and *ReferenceSystem*. Though the *PhysicalPlant* and *PhysicalReference* appear in this aspect, they represent physical entities rather than control design components.

4.4.3 Structural Semantics

The main design goal of PaNeCs is to ensure "correctness-by-construction" for passive NCS. To achieve this objective we impose constraints on the NCS component properties as well as their interconnections. The language semantics require constraints that cannot be completely captured with only the metamodeling notations above. Using the Object Constraint Language (OCL), we can specify well-formedness rules to precisely control the static semantics of the language. GME is embedded with an OCL engine such that specified constraints are enforced at design time, giving direct feedback whenever the user attempts to create unacceptable connections in the model or violate any other specified constraints.

Three classes of constraints are implement towards to the structural correctness of PaNeCS models: *Cardinality Constraints* ensure that the correct number of components are used in the NCS design. For example, for each *PlantSystem* model there must be one *Plant*. *Connection Constraints* restrict the number of allowable connections between components. For example, in the *PlantSystem* model there can only be one bidirectional connection between the *Plant* and *BilinearTransformP*. *Unique Name Constraints* ensure the uniqueness of the names of components in the Plant and Controller subsystems as well as in the top level model of the NCS. The sample OCL constraint below specifies that the number of connections from a BilinearTransformC model to a DigitalController must be at most one.

```
Description: There must be only one bidirectional connection between...
BilinearTransformC to the DigitalController
Equation: let dstCount = self.attachingConnections("src",Controller_Bilinear)...
->size indstCount <> 0 implies dstCount = 1
```

4.5 Passivity Analysis

In the typical construction of NCS, the system design needs to be analyzed in order to ensure they satisfy specified system properties and as the model of the system increases in size this process becomes increasingly difficult and tedious and maybe even intractable. In passivity-based NCS, in order to ensure overall system passivity, the system components need to be analyzed to ensure they satisfy passivity constraints. Using an integrated analysis tool in PaNeCS, we can currently analyze the passivity properties of any LTI system component which can be specified in state space representing the model of a plant or controller.

4.5.1 Component Analysis

The analysis of LTI plants and controllers specified in PaNeCS is performed automatically using an integrated Matlab analysis function. Since each of these components is assumed to have a LTI state space representation characterized by the matrices A, B, C, D of compatible sizes, we use Linear Matrix Inequalities (LMIs) together with the CVX semidefinite programming tools for Matlab [312] [313]. For example, a LMI formula for strict output passivity for an LTI discrete plant or controller is given by

$$\begin{bmatrix} A^TPA - P - \hat{Q} & A^TPB - \hat{S} \\ (A^TPB - \hat{S})^T & -\hat{R} + B^TPB \end{bmatrix} \leqslant 0$$

$$\begin{aligned} \hat{Q} &= C^TQC, \quad \hat{S} = C^TS + C^TQD \\ \hat{R} &= D^TQD + (D^TS + S^TD) + R \end{aligned} \tag{51}$$

$$\exists \varepsilon > 0,\ Q = -\varepsilon I,\ R = 0,\ S = \frac{1}{2}I, \exists P = P^T > 0$$

The CVX semidefinite programming (SDP) tool is used in a Matlab script to solve the LMI for each component. The analysis tool, on invocation (i.e. the modeler presses a button), executes a C++-based model interpreter within GME which traverses a PaNeCS model visiting LTI components, and invokes the analysis function to test the passivity of each LTI component in the model. After each test, the analysis tool notifies the user whether the component passes or fails the passivity constraint test. Following this process, a user can go back and change the components of the model that fail the passivity test.

4.5.2 System-Level Analysis

Due to the "correct-by-construction" approach used in our framework in designing networked control, we only analyze the LTI plants and controller components for passivity. If these components satisfy the passivity constraints, the network control system as a whole also satisfies the passivity principles. The component interconnections are restricted in such a way that they are "correct-by-construction". Only valid connections are allowed amongst components, so any interconnected system of passive components in the language will be globally passive. Therefore, the analysis tool together with the passivity-based constraints encoded in the semantics of the modeling language, greatly reduces the analysis burden for determining passivity and hence stability of the composed NCS design. Using this approach, passivity analysis of NCS, such as the NCS architecture in Figure 11 provided in Section 4.2, can be easily performed.

4.6 Code Generation

In order to evaluate NCS designs in PaNeCS, we developed model interpreters that can be used to synthesize software for integration, deployment and testing of NCS. We developed two model interpreters, one for generating models for the purpose of simulations and the other for running actual experiments.

(i) Generation of Simulation Models

This model interpreter synthesizes simulation models from PaNeCS NCS models. The interpreter is developed in C++ using the Builder Object Network (BON2) API provided with GME [310]. The interpreter traverses all the entities of a particular networked control system instance model and extracts the model components and corresponding parameters. These parameters and model structure are used to generate MATLAB files for configuring and building Simulink and TrueTime models to simulate the NCS. The *PlantSystem* and *ControllerSystem* are modeled as Simulink subsystems, which contain the respective *Plant* and *DigitalController* behavior blocks. Each generated system-level component is connected to a TrueTime Kernel block. The TrueTime Kernel models a processing node with a scheduler and I/O. Our models execute on periodic schedules within TrueTime. For this version of our language, the *PowerJunction* is implemented as a task in the TrueTime Kernel connected to the *ControllerSystem*. The task that implements the power junction is triggered by data arrival events. Each TrueTime kernel has an initialization script and a function script specifying timing for I/O and task execution. The TrueTime Network and TrueTime Wireless network blocks simulate the transfer of data packets over a network from one node to another.

(ii) Generation of Executables for Experiments

Similar to the case of synthesizing simulation models, we developed a code generator that can be used to synthesize software for performing experimental evaluation of NCS. The code generator is also developed in C++ using the Builder Object Network (BON2) API. For the purpose of experiments, the Simulink models alone are not sufficient to set up the network infrastructure. The deployment model, which can be visualized through the *Processor Assignment Aspect*, describes assignments of models to processors, so we also generate bash scripts from the deployment model in order to set up and run the experiments. The scripts handle the network configuration on each node, and then start up the model with the proper parameters. The network infrastructure utilizes Netcat and SSH. The power junction, is configured with a group of servers, and the plant and controller models use client sockets to attach to the power junction. The client connections transmit data over TCP through a secure shell tunnel to the power junction. This technique hides many of the network configuration details from our setup. The Netcat instances serve the purpose of adapting socket types (i.e. client to server and TCP to UDP) as well as making the Simulink socket connections more robust to failures. A typical script sets up the SSH tunnel first, runs Netcat to adapt the sockets, and then runs the model using MATLAB. When the model comes up it finds all of the necessary socket connections available, whether or not the other models have started yet.

In this framework, the networked system follows a globally asynchronous locally synchronous execution model. Components are executed locally in a synchronous manner based on the local sampling period. The controller and plant receivers execute periodically. To preclude the possibility of blocking, zeros are supplied for missing data values to avoid introducing energy into the system, thereby preserving passivity. It is assumed that data messages will not arrive out of order. Our implementation uses secure TCP links between PCs. Except in extreme overload conditions, message order is maintained.

In regards to buffer sizing, all data supply and consumption rates are known and adequate for nominal operation. As long as the PC clocks remain relatively close to each other, buffers will never grow without bound. However, currently we do not provide any guarantee for non-ideal operation. If a message receiver crashes or is otherwise delayed, then the sender could continue to fill the intervening buffer indefinitely which can potentially lead to a crash). We can handle this contingency by having the sender drop unsent data instead of storing it.

4.7 Simulation Results

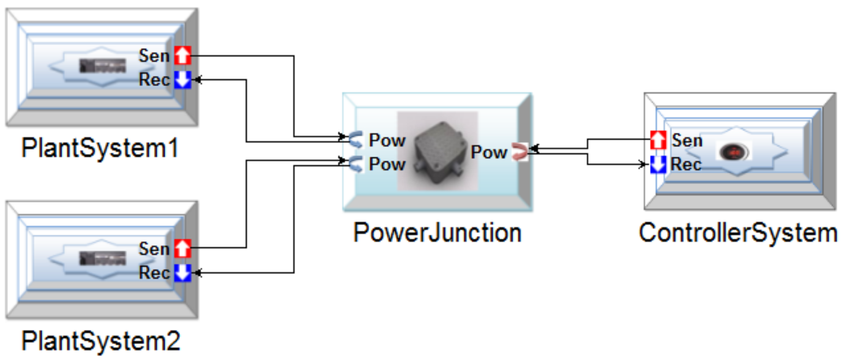
In this section, we present the simulation of a networked control system composed of a linear controller and two discrete-time linear plant systems. We show that networked control systems designed using PaNeCs are robust and remain stable when subjected to uncertain network effects. The controller controls the plants to simultaneously track a specified trajectory. The overall NCS is modeled in PaNeCS and subsequently Matlab/Simulink and TrueTime code is automatically generated from the models for evaluation. Although this case study models two discrete-time plants, PaNeCS can model and simulate an arbitrary number of plants. Figure 20 shows a model for a simple passive linear controller regulating two passive linear plants to track a specified reference signal. Figure 20a denotes the *Control Design Aspect* describing the control design concepts while Figure 20b denotes the *Platform Aspect* describing the physical platform components.

The two plants in the model, shown in Figure 20, are simple integrators (corresponding to models of inertial masses of 2kg and .25kg respectively) which are discretized in time. The plants' dynamics were modeled in state space form and the corresponding A, B, C, and D matrices as well as sampling time (T_s) were provided as parameters to the instance model.

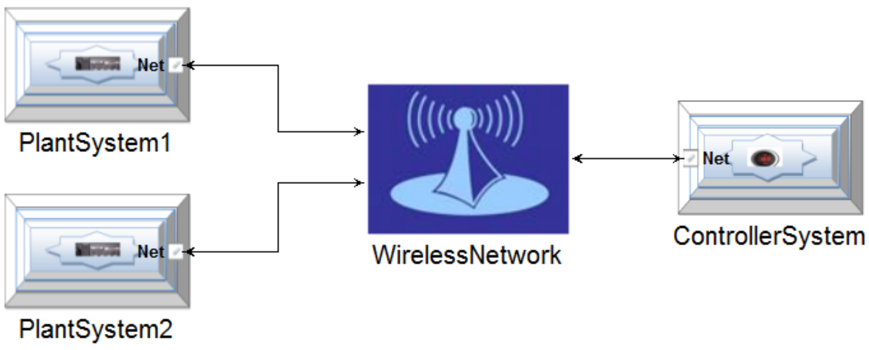
The controller for controlling the plants are also specified in the state-space form (A, B, C, D, and T_s). The parameters values for the plant and controller dynamics are given in Table 1. The analysis tool checked and verified that the LTI plants and the controller models satisfy the passivity constraints. Then the integrated PaNeCS model interpreter for generating simulation code was used to generate platform-specific Simulink simulation models based on the specified parameters and model structure.

4.7.1 Nominal Conditions

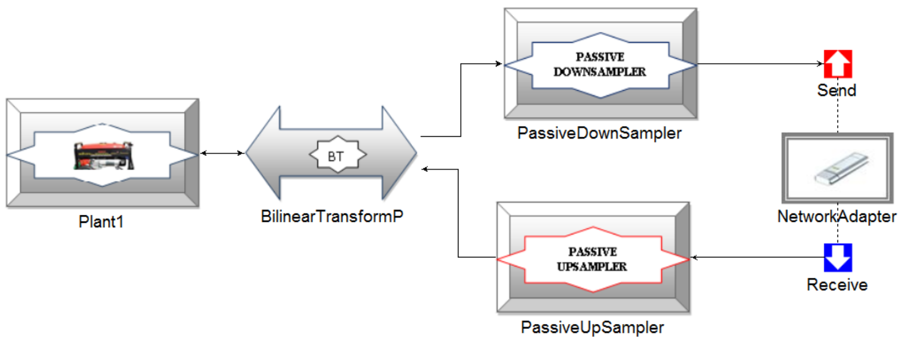
In this scenario, we consider nominal network conditions, hence no addition disturbances were introduced into the network. Figure 21a shows that the plants closely tracked the reference velocity.



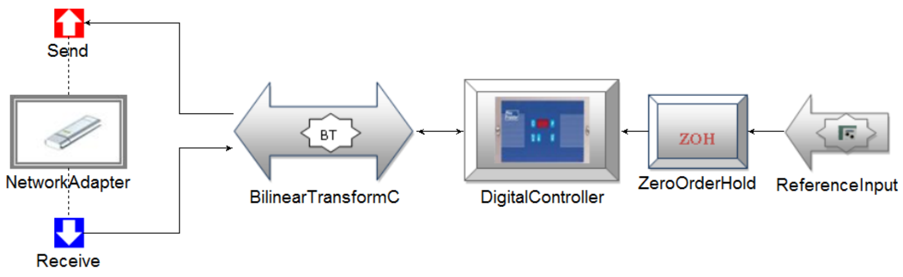
(a) Control Design Aspect



(b) PlatformAspect



(c) Plant Subsystem



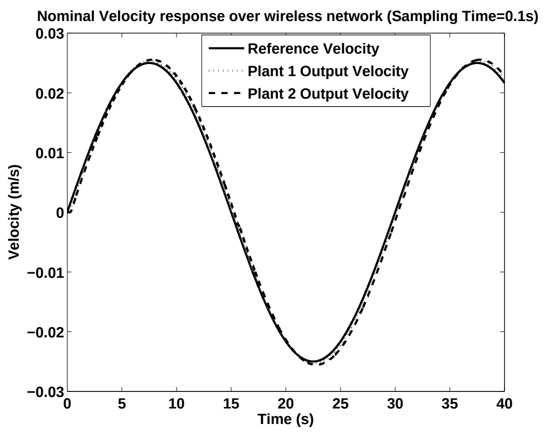
(d) Controller Subsystem

Figure 20: Sample Model of a Networked Control System (Linear Plants)

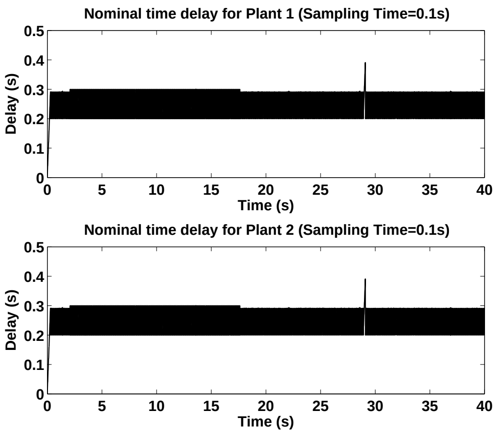
The round trip delay for each plant seemed to have very little effect on the stability of the plants' velocity response. The delay as seen in Figure 21b, can be attributed to the internal processing of the plants and controllers rather than network delay itself.

Table 1: LTI Plant and Controller Dynamics.

	A	B	C	D	T_s
Plant1	1	1	.005	.0025	$.01s$
Plant2	.996	1	.04	.02	$.01s$
Controller	0	0	0	10π	$.1s$



(a) Nominal velocity response.



(b) Time delays .

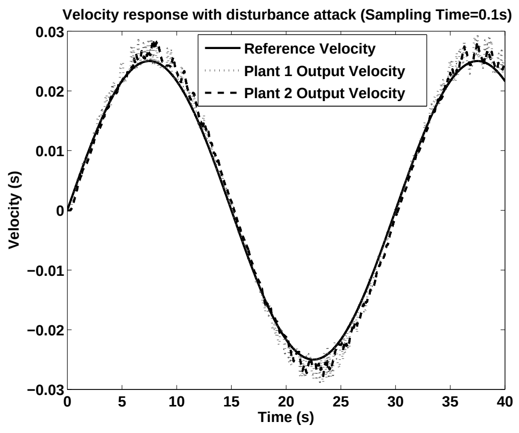
Figure 21: Velocity and Delay Plots (Nominal).

4.7.2 Network disturbances

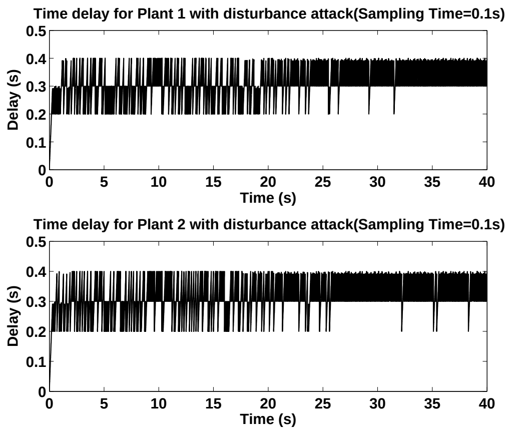
This experiment introduces a disturbance attack in the network using parameters on the wireless network block. Figure 22a and 22b shows the velocity response and the time delay respectively for each plant. The results show that even with disturbance attacks, the plants remain stable in tracking the reference velocity although the performance is relatively affected as can be seen from the plots. This demonstrates the advantage of the passivity approach which guarantees the stability of the NCS in the presence of uncertainties due to network effects.

4.8 Experimental Results

In this section, we present the experimental results for a Networked Multi-Robot System (NMRS) modeled using PaNeCS. The instance model for the NMRS is designed in PaNeCS specifying the



(a) Velocity response with disturbance attack.



(b) Time delays with disturbance attack.

Figure 22: Velocity and Delay Plots (Network Disturbance).

components of the NCS. Subsequently, the software components and deployment model required for executing the actual NMRS are automatically generated and deployed on the platform.

4.8.1 Experimental Setup

The experimental setup consists of two CrustCrawler robotic arms [314] and one Novint haptic paddle [315] connected using a networked computing platform. The computing platform consists of five networked Windows PCs with Matlab/Simulink. The robotic arms and the haptic paddle are connected to three different PCs via USB interface utilizing Matlab/Simulink APIs. The two additional PCs are used to implement various software components of the networked architecture. The distributed platform is modeled in PaNeCS (see Figure 27). The CrustCrawler robot shown in Figure 23 has four degrees of freedom with AX-12 smart servos at each joint [314]. Each of these servos has three inputs and five outputs. The inputs are position, velocity, and maximum torque value, and the outputs are actual position, actual velocity, temperature, load, and feedback voltage. Joints one and four contain one servo while joints two and three each contain two servos for a total of seven servos per robot. The robot can be represent as having four points of mass located at the mid-point of each link. The point masses are: $m_1 = 0.362$ kg, $m_2 = 0.401$ kg, $m_3 = 0.059$ kg, and $m_4 = 0.177$ kg.

The Novint haptic paddle shown in Figure 25 provides the desired trajectory to be tracked by the robotic arms in a synchronized fashion. The paddle requires the Haptik Library [315] that provides

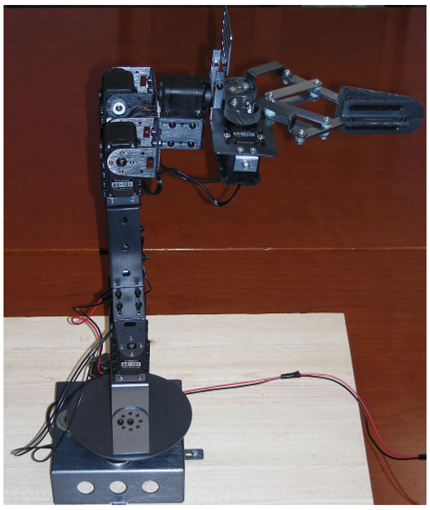


Figure 23: CrustCrawler 4 DOF Arm

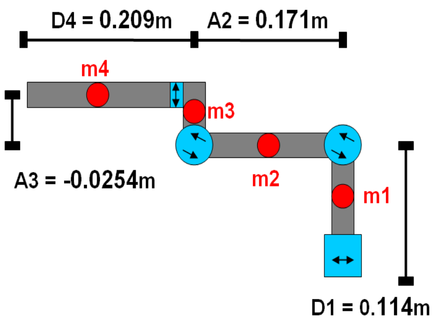


Figure 24: CrustCrawler Model

an interface between most haptic paddles and various computer languages like C/C++, Java, and MATLAB. The haptic paddle API includes forward kinematics software that transforms the joint positions of the three legs into *x*-*y*-*z* coordinates. When the paddle end effector is moved by the user, a position signal in *x*-*y*-*z* coordinates is sent into a Simulink haptic paddle block.

4.8.2 PaNeCS Model of the Networked Multi-Robot System

In order to illustrate the model-based framework, we present the NMRS model developed in PaNeCS.Figure 26 shows the *Control Design Aspect* that visualizes the control design modeling layer and Figure27 shows the *Platform Design Aspect* which shows the physical components of the system as well as their interconnections. Our design is based on the assumption that each of the top level models, which include the *PlantSystem*, *ControllerSystem*, *PowerJunction* and the *ReferenceSystem* is implemented on a separate processor. The mapping of these components to respective processors is

Figure 25: Novint Haptic Paddle

Table 2: PlantSystem Model Parameters

Parameter	Value
T_s	0.04s
q_0	[0 -π/2 π/2 0]
K_v	0.15
τ	2
Samplesize	2
b	2
InputVector, N	4

performed by the *ProcessorAssignmentLayer* of the model (that is not shown here).

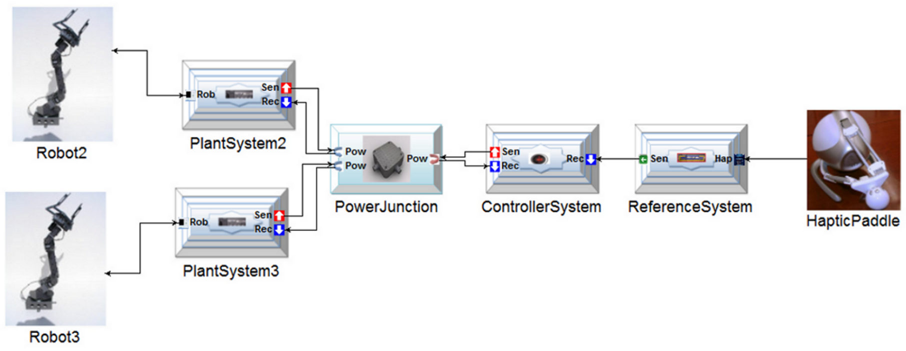


Figure 26: Control Design Layer

The details for implementing the *PlantSystem* which is identical for the two robots are shown in Figure 28 and the required parameters are listed in Table 2. The components for implementing the *ControllerSystem* are also shown in Figure 29 and the controller parameters are in Table 3. Finally, Figure 30 shows the components for implementing the *ReferenceSystem*.

After configuring and entering the desired parameters for the NMRS model, the code generator is used to generate Simulink models and network scripts. The Simulink models and network scripts

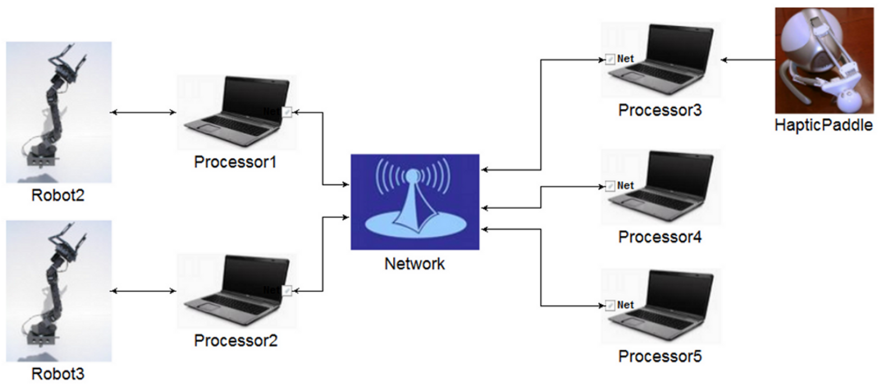


Figure 27: Platform Design layer

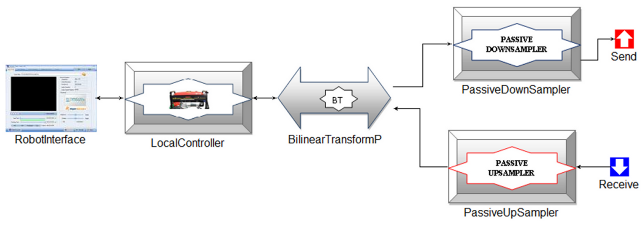


Figure 28: Plant Sub-system

are then deployed on the respective PCs based on the deployment model.

4.8.3 Results

4.8.3.1 Experiment 1: Nominal Case

In this experiment all PCs communicate through a 100BASE-TX Ethernet network. Initially, the PCs are not connected to the network and the connection sequence is the power junction, robot 2, robot 3, and controller. Figure 31 shows the x-, y-, and z-coordinates of the robots, along with the

Table 3: Controller Model Parameters

Parameter	Value
T_s	0.08s
K	0.5
w	$\frac{\pi}{2}$
w_n	$2\frac{\pi}{10}$
ζ	0.92
b	2
InputVector, N	4

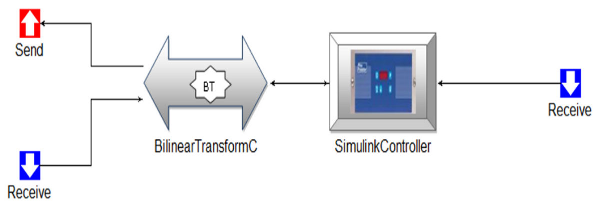


Figure 29: Controller Sub-system

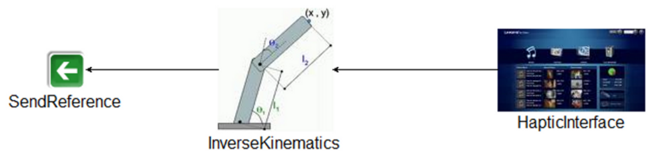


Figure 30: Reference Sub-system

reference trajectory, and also the angular position of joint 2 of each robot and the respective reference trajectory. Each robot initially adjusts to its home position. Once the controller is connected to the power junction, the robots begin following the reference trajectory. The robots track the reference trajectory in a synchronized manner. The trajectories of the robots are similar enough to be almost indistinguishable in the plots.

If there is packet loss, the components assume zeros so that no energy is introduced and the system remains passive. Note that other design decisions that preserve passivity can be used. Because of the zeros, the robots attempt to return to their home position, causing the robots to jerk while following the reference trajectory. To compensate for this undesirable behavior, a low-pass filter is added to eliminate this high frequency motion. The trade off is attenuation and smoothing of the trajectory. Despite this, the robots are well synchronized and the NMRS is stable.

To show the network delay, we plot the first dimension of the wave variable transmitted from robot 2 to the power junction and back in Figure 32. The delay is time varying and on the order of one second due to the network link, and the buffering of data in the Netcat and SSH components.

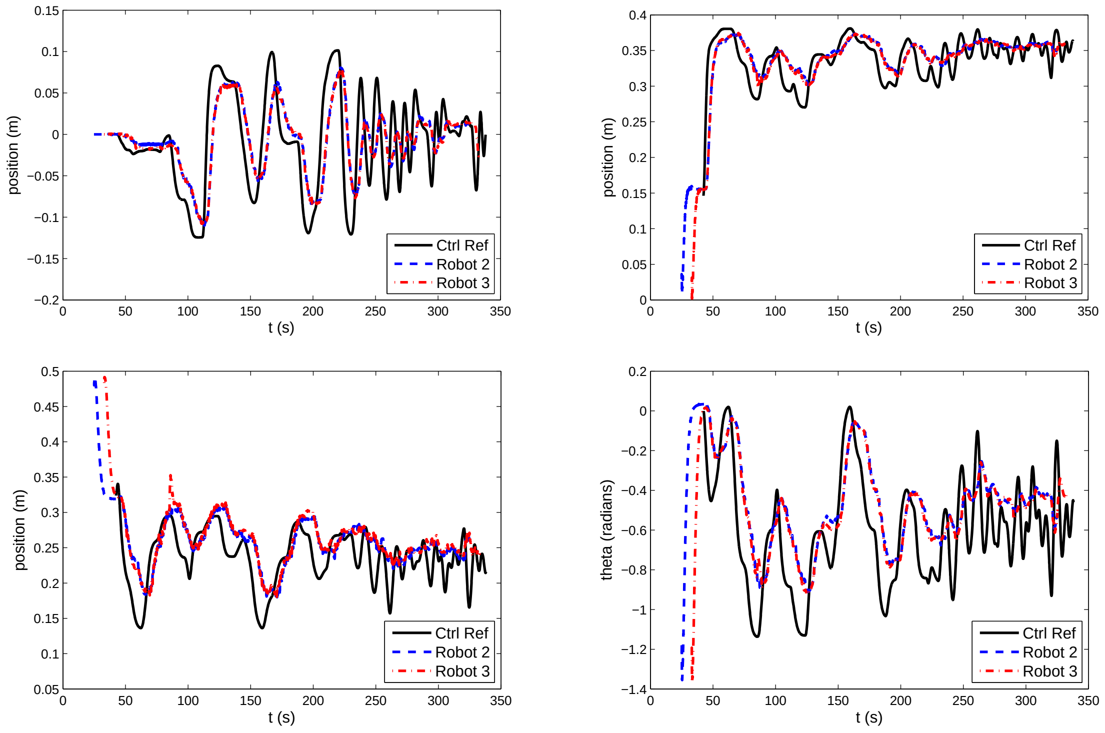


Figure 31: x-y-z coordinates and angle of joint 2 of reference, robot 2, and robot 3.

4.8.3.2 Experiment 2: Persistent Link Interruptions

Experiment 2 demonstrates the robustness of the NMRS to persistent link interruptions. To emulate the link interruptions, a boolean variable is implemented in each plant and the controller, which controls the data flow to the power junction, and can be toggled online. While the link is interrupted, the power junction simply sends zeros to the respective component. In this way, we avoid shutting down and restarting the network infrastructure that requires considerable time. During the experiment, robot 3 is interrupted, then robot 2, and finally, the controller.

The angular position of joint 3 of both robots and the reference trajectory are shown in Figure 33 to illustrate how each robot returns to its home position while its link to the power junction is interrupted. The figure also shows the y-coordinate of each robot with the reference trajectory. Since the connection sequence is identical to the nominal experiment, the order in which the robots are first seen is identical, and each initially adjusts to its home position. Once the controller is connected to the power junction, the robots begin following the reference trajectory until its link is interrupted. After approximately 52 sec the data flow of robot 3 is interrupted with the power

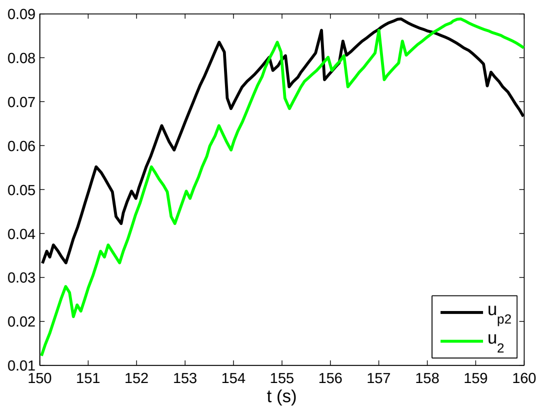


Figure 32: Time Delay Between Robot 2 and Power Junction

junction. While interrupted, each of its servos returns to its home position. At approximately 75 sec robot 3 is reconnected and resumes following the reference trajectory. Next, around 95 sec, the data flow of robot 2 is interrupted with the power junction. Identically as before, the joints return to the home position while interrupted. Robot 2 is reconnected at approximately 118 sec. Finally, the data flow of the controller is interrupted with the power junction around 151 sec, causing both robots to return to their home position. Again, once the controller is reconnected at approximately 180 sec, the robots resume following the reference trajectory. Since the link interruptions are implemented in a very controlled manner, no additional network delay is caused by the link interruptions and the delays are similar to the nominal case.

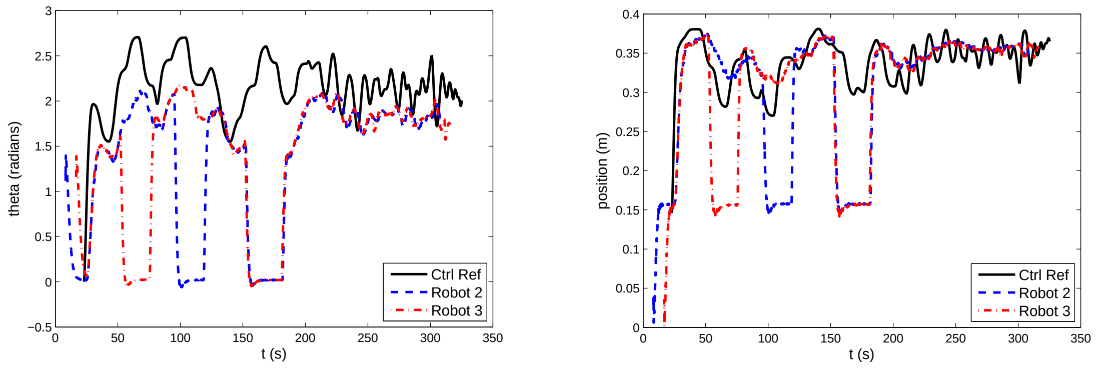


Figure 33: Persistent link interruptions: Angle of joint 3 and *y* coordinate of reference, robot 2, and robot 3.

4.8.3.3 Experiment 3: Intermittent Wireless Connection

In Experiment 3, the PC running the controller is connected to the Ethernet network through an 802.11b wireless connection. The experiment demonstrates the performance of the NMRS with an intermittent communication channel. Figure 34 shows the angular position of joint 3 of the robots along with the reference trajectory. Again, each robot initially adjusts to its home position. Due to the increased network delay, the plants do not begin tracking the reference trajectory until after approximately 29 sec into the experiment.

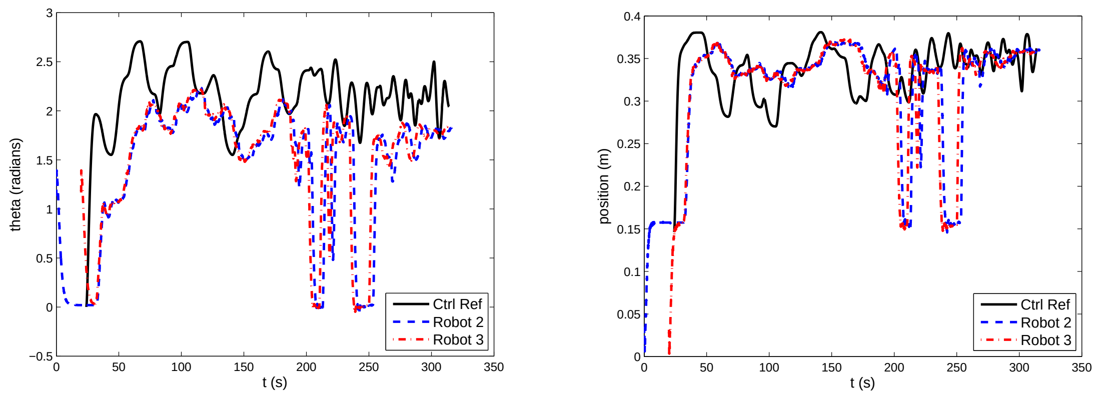


Figure 34: Intermittent connection: Angle of joint 3 and y coordinate of reference, robot 2, and robot 3.

Because of the unreliable wireless connection between the controller and power junction much data is dropped, causing more high frequency components in the trajectories of the robots. Even with the low-pass filter, the trajectories are clearly not as smooth and accurate as in previous experiments. Observe, for example, the peak between 35 and 40 sec in Figure 33 and compare it to the behavior in Figure 34. Obviously data is being dropped from the controller during this time since in both robots there is more attenuation and no well defined peak as in Figure 33. Between 204 and 260 sec the connection gets significantly worse. Despite the intermittent connectivity, the NMRS remains stable and still manages to track the reference reasonably well while the connection is established. Again, the y-coordinate of each plant, along with the reference trajectory, is shown in Figure 34.

4.9 Summary

Our model-based approach simplifies the process of designing “correct-by-construction”’ passive networked control systems. The integrated analysis tool, together with encoded passivity constraints in the modeling language significantly reduces the burden typically involved in analyzing NCS for stability. The integrated model interpreters facilitate rapid prototyping of passivity-based NCS, allowing for quick model reconfiguration, code generation and evaluation using simulations or actual experiments. We present simulation results on the networked control of linear plants as well as experimental results on a networked multi-robot system. The presented results illustrates the effectiveness of our approach as well as the robustness of NCS modeled in PaNeCS.

CHAPTER 5

PASSIVITY-BASED ADAPTIVE SAMPLING CONTROL

5.1 Introduction

In the traditional NCS, the computation and exchange of information over the communication network is performed periodically based on a fixed sampling interval. This periodic strategy allows the use of well-established theory in sampled-data systems to design and analyze the overall system [316][114]. Although, the periodic strategy is well-known and convenient, it is overly conservative, and can be categorized as "open-loop sampling" which essentially implies the periodic or static use of the computational and network resources regardless of the conditions of the network or changes in the system that is being controlled i.e. the sampling approach is not controlled by any feedback mechanism. An alternative strategy, adaptive sampling, has become very important due to the strong need for the efficient use of limited computational and network resources [116][317],[318],[115]. The main idea of adaptive sampling is to compute the sampling interval, T_i, at which a control law needs to be executed or when an information needs to be sent over a network, based on a specified condition. As outlined in Chapter 2, there are only a few results addressing adaptive sampling in NCS framework. In NCS, the variability of the sampling intervals in adaptive sampling poses additional challenges on the achievable stability and performance of the NCS especially in the presence of network uncertainties such as delays and packet losses.

In this chapter, in an effort towards efficiently utilizing network resources while achieving performance objectives in an unreliable communication network, we propose an integrated NCS architecture to tackle the two highlighted challenges involving limited network resources and network effects. Specifically, we integrate the concept of passivity and adaptive sampling in order to achieve both the resilience of NCS in the presence of network effects as well as the efficient utilization of the limited network resources. The integration is used in a hierarchical NCS [15] framework for the purpose of trajectory tracking. The proposed framework is general in the sense that it can be adapted for other control performance objectives but for clarity we present the framework for a trajectory tracking objective. The integrated framework is performance-aware in the sense that the utilization of the resources is based on achieving a desired tracking performance. Our approach differs from most of the existing approaches which are typically designed for the main purpose of guaranteeing stability. The integrated adaptive sampling framework not only ensures the efficient utilization of network resources but also provides the flexibility of incorporating other adaptive sampling approach for example network scheduling adaptation.

The main contribution of this work is the integration of two fundamental control theoretic concepts, passivity and adaptive sampling, in order to simultaneous address the challenges of robustness to network effects as well as efficient utilization of network resources. We present an NCS architecture that highlight the integration. We introduce a pair of sample-and-hold components, a variable passive sampler and variable passive hold, that handles non-uniform sampling intervals while guaranteeing passivity. We analytically demonstrate passivity of the proposed NCS architecture. Also, we analytically demonstrate the performance objective of trajectory tracking in the hierarchical NCS framework commonly used in robotics and unmanned aerial vehicles (UAV) applications. Finally, we present both simulations and experimental results involving a case study on the trajectory tracking of a robotic arm over a network in order to demonstrate the benefits of the integrated NCS architecture and the effectiveness of the approach.

The rest of the chapter is organized as follows: The problem addressed in the chapter is clearly formulated in Section 5.2. The proposed NCS architecture is introduced in Section 5.3. The analytical results on passivity and trajectory tracking are presented in Section5.4. We provide simulations and experimental results for a robotic tracking case study in Sections 5.5 and 5.6 respectively. Finally, a summary is provided in Section5.7.

5.2 System Model and Problem Statement

This chapter considers the problem of achieving trajectory tracking in an unreliable communication network with limited resources. First, we describe the notations used in the following sections. The notations used throughout are standard. The set of natural numbers, integers, and real numbers are denoted by $\mathbb{N} = \{1, 2, \dots\}$, $\mathbb{Z}$, and $\mathbb{R}$, respectively. The m-dimensional Euclidean space is $\mathbb{R}^m$ and the set of all m by n matrices over the real numbers is $\mathbb{R}^{m\times n}$. The transpose of a vector $x \in \mathbb{R}^m$ and matrix $M \in \mathbb{R}^{m\times n}$ are given by x^T and M^T respectively.

5.2.1 System Model

In this section, we describe the system model as well as the underlying assumptions. We consider the system model,H_{mp}, as depicted in Figure 35. The main components of the model are the dynamical plant system, H_p, the local controller, H_{lc} and an error function block, H_{es}.

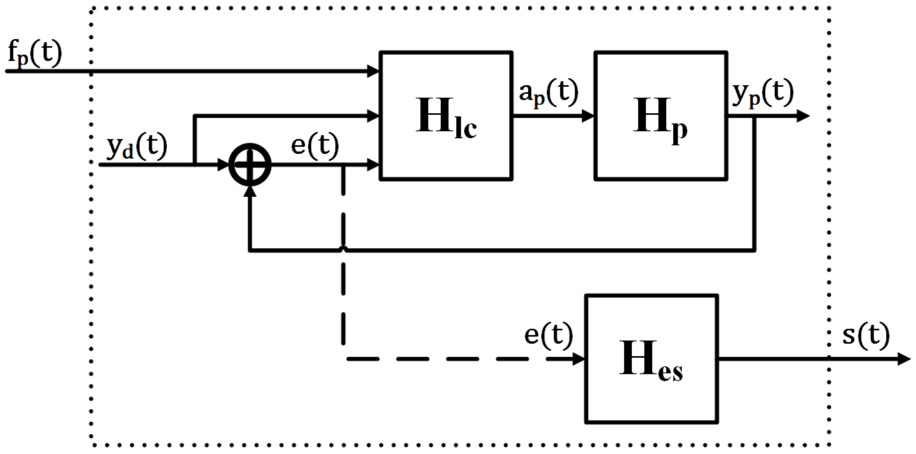


Figure 35: System Model, H_{mp}.

(a) **Plant Model**: The dynamical plant, H_p, which could be linear or nonlinear is defined as

$$H_p : \begin{cases} \dot{x} = f(x, a_p) \\ y_p = h(x) \end{cases} \tag{52}$$

where $x \in \mathbb{R}^n$ represents the state variables, $a_p \in \mathbb{R}^m$ represents the control inputs to the plant and $y_p \in \mathbb{R}^m$ represents the plant outputs. The variable, e, is the error defined as the difference between the plant output, y_p and a reference trajectory, $y_d \in \mathbb{R}^m$.

(b) **Local Controller**: The local controller, H_{lc}, in Figure 35, is a functional block which takes f_p, y_d and e as inputs and computes an output a_p which is fed into the plant.

(c) **Error Function**: The functional block, H_{es}, is an error operator which relates the error, e, to the output, s as shown in Figure 35.

We assume the following about the system model, H_{mp}

Assumption 1: The input-output mapping, from input, f_p to output, s as shown in Figure 35 can be defined such that $H_{mp} : f_p \mapsto s$ is passive.

Assumption 2: H_{mp} is zero-state detectable [303]. This assumptions ensures that passive mapping $H_{mp} : f_p \mapsto s$ is stable.

Assumption 3: There exists a function, H_{es}, relating e to s with Λ defined as a positive diagonal matrix, in the form,

$$s = \dot{e} + \Lambda e \tag{53}$$

such that if $s \to 0$ then

$$\lim_{t \to \infty} e(t) = 0 \tag{54}$$

.

Assumption 4: The desired reference trajectory, y_d, is bounded and twice differentiable.

In regards to the communication over the network, we assume the following on the packet handling of network messages:

Assumption 5: The networked packet handling mechanism is designed to ensure that no duplicate packet is processed.

Assumption 6: In the case of packet loss, if the input buffer is empty, null packets are processed.

Assumptions **5** and **6** ensure that no additional energy is introduced by the communication channel.

5.2.2 Problem Statement

The two main problems considered in this work are described as follows:

1. *We consider the problem of designing a robust hierarchical networked control system for achieving the following tracking objective in an unreliable wireless network:*

   (a) *The output, y_p, of the plant, H_p, tracks a local reference trajectory, y_d such that*

   $$\lim_{t \to \infty} y_p(t) - y_d(t) = 0 \tag{55}$$

   (b) *In the presence of a non-local reference input, r_c, which can be used to model behaviors such as the presence of an obstacle in the plant's environment, the plant output tracks a modified reference signal, y_{da} such that*

   $$\lim_{t \to \infty} y_p(t) - y_{da}(t) = \lim_{t \to \infty} y_p(t) - y_d(t) - r_c(t) = 0 \tag{56}$$

Hence, ensuring that the plant avoids the obstacle.

2. *We consider the problem of efficiently utilizing the network resources while ensuring that the tracking objective is achieved.*

In the following sections, we propose a solution to tackle the above problems.

5.3 Integrated Passivity-based Adaptive Sampling Control Architecture

Figure 36, depicts the proposed networked control architecture which we refer to as passivity-based adaptive sampling control (PBASC) architecture . This architecture is designed to achieve trajectory tracking as well as the efficient use of network resources. The remote system and the networked controller are shown on the left and right sides of the network in Figure 36 respectively. On the remote system side of the network, the block H_{mp} represents the system model introduced in Figure 35. In what follows we describe the components of Figure 36.

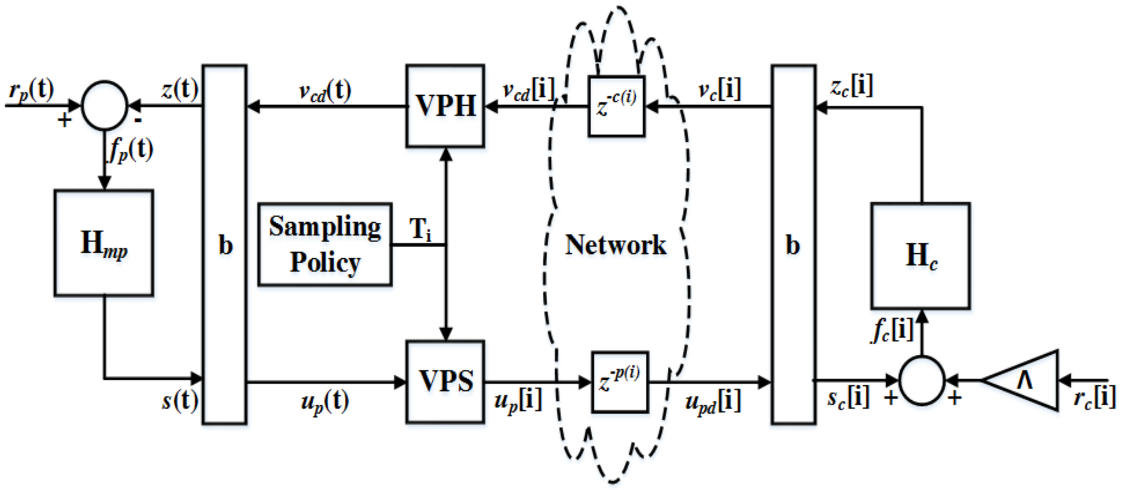


Figure 36: PBASC Architecture.

5.3.1 Adaptive Sampling Scheme

The sampling policy box, in Figure 36 depicts an adaptive sampling scheme which outputs the sampling intervals. In order to interconnect the system, H_{mp} to the digitally implemented networked controller, a sample-and-hold mechanism is needed. Traditionally, the sample-and-hold mechanism is based on fixed sampling periods but in order to efficiently utilize network bandwidth we use an adaptive sampling scheme. The adaptive sampling mechanism can be designed based on any of the existing adaptive sampling techniques in order to generate sampling intervals. For the purpose of tracking, we need a mechanism that determines the sampling intervals based on a function of tracking error. We chose the adaptive sampling mechanism based on the self-triggered control concept described in [319]. This approach enables us to specify adaptive sampling scheme in terms of a storage function, a lyapunov-like function of the tracking error.

Our triggering mechanism is based on the system model, H_{es}, relating the tracking error, e, with the output of H_{mp}, s. Rearranging and restating (53), we obtain the system model defined as

$$\dot{e}(t) = -\Lambda e(t) + s(t); \tag{57}$$

$$s(t) = s(t_i), \quad t \in [t_i, t_{i+1}) \tag{58}$$

where $[t_i]_{i\in\mathbb{N}}$ is an increasing sequence of sampling times with $t_0 = 0$. This model represents a strictly output passive mapping with the tracking error, e, as the output and s, as the input. The system model in (57) and (58) could be represented in a minimal state-space realization, where the tracking error, e, is the state as well as the output of the system and s, is the input. This results in the state-space coefficients where $A = -\Lambda$, $B = C = I$, the identity matrix, and $D = \mathbf{0}$.

A map, Γ_d, can then be used to define a self-triggered implementation of the system model given by (57) and (58). This map determines the t_{i+1}, as a function of the tracking error output, e at the time t_i, i.e., $t_{i+1} = t_i + \Gamma_d(e(t_i))$. If we denote by T_i, the sampling interval $T_i = t_{i+1} - t_i$ we have $T_i = \Gamma_d(e(t_i))$. As described in [319], the design of a self-triggered policy involves a sequence of steps.

First, an output function to describe the evolution of the system's tracking error needs to be determined. The output function for our sampling policy is the storage function of the system defined by (57) and (58).

$$V(e) = e^{\mathsf{T}} P e \tag{59}$$

where P is a positive definte matrix satisfying the passivity constraints for the system model defined by (57) and (58). The passivity constraints for the strictly output passive system is defined by the following Linear Matrix Inequality (LMI) constraints [296].

$$\begin{bmatrix} A^TP - PA - \hat{Q} & PB - \hat{S} \\ (PB - \hat{S})^T & -\hat{R} \end{bmatrix} \leqslant 0$$

$$\hat{Q} = C^TQC, \quad \hat{S} = C^TS + C^TQD \tag{60}$$
$$\hat{R} = D^TQD + (D^TS + S^TD) + R$$

$$\exists \varepsilon > 0,\ Q = -\varepsilon I,\ R = 0,\ S = \frac{1}{2}I$$

The output function defined in (59) has an estimated decay rate, ρ_0 which can be computed from the parameters of (60) as

$$\rho_0 = \frac{\gamma_{min}(Q)}{\gamma_{max}(P)} \tag{61}$$

where γ_{min} and γ_{max} represent the minimum and maximum eigenvalues respectively of the indicated matrices [320].

Next, we define a performance specification in terms of the tracking error output. Our performance function is an exponentially decaying function of the output function with an initial value as the current sampled error. The performance specification is defined as:

$$D(t) = V(e(t_i))exp^{-\rho(t-t_i)} \tag{62}$$

In order to guarantee that the performance function bounds the output function, the decay rate of the performance specification is chosen as $\rho < \rho_0$.

With the output function and performance specification, we determine a continuous time triggering function. From (59) and (62), the triggering condition is given as

$$h_c(t_i, e(t), e(t_i)) := V(e(t)) - V(e(t_i))exp^{(-\rho(t-t_i))} \leq 0; \tag{63}$$

for some $0 < \rho < \rho_0$.

Finally, a self-triggered policy can then be determined from the continuous time triggering function. In order to check when the triggering condition defined in (63) is violated, we consider a discrete-time implementation based on a discrete step size, $\Delta \in \mathbb{R}^+$ since no continuous time implementation can be used to perform this check. With, T_i defined as the sampling interval, let T_{min} and T_{max} be defined as the minimum and maximum sampling intervals respectively. Also $N_{\text{min}} := \lfloor T_{\text{min}}/\Delta \rfloor$, $N_{\text{max}} := \lfloor T_{\text{max}}/\Delta \rfloor$. The discrete-time implementation can be defined as follows:

$$h_d(e(t_i), n) := h_c(t_i, e(t), e(t_i)) \quad \forall n \in [0, N_{\text{max}}] \quad \forall i \in \mathbb{N} \tag{64}$$

From this discrete-time implementation, the map $\Gamma_d : \mathbb{R}^n \mapsto \mathbb{R}^+$, for computing the next sampling interval or time for the tracking error system model given in (57) is given by:

$$\Gamma_d(e(t_i)) := \max\{T_{\text{min}}, n_i \Delta\} \tag{65}$$

where

$$n_i := \max_{n \in \mathbb{N}} \{n \leq N_{\text{max}} | h_d(e(t_i), c) \leq 0, c = 0, ..., n\} \tag{66}$$

The lower and upper bounds of the sampling intervals are explicitly enforced by T_{min} and T_{max} respectively. The upper bound enforces robustness of the implementation and limits computational complexity. The discrete time step, Δ, can be chosen based on desired accuracy and computational complexity.

5.3.2 Wave Variables

The blocks denoted as **b** in Figure 36 each represents the wave variable transformation which was previously introduced and described in Section3.6. On the left hand side of Figure 36, the control signal, $z(t)$ and the system's output signal, $s(t)$ are transformed into the wave domain through the scattering transformation. The scattering transformation produces the continuous wave variables $u_{\mathsf{p}}(t)$ and $v_{\mathsf{cd}}(t)$, which are related to the signals, $z(t) \in \mathbb{R}^m$ and $s(t) \in \mathbb{R}^m$ as follows:

$$\frac{1}{2}(u_p^{\mathsf{T}}(t)u_p(t) - v_{cd}^{\mathsf{T}}(t)(t)v_{cd}(t)) = s^{\mathsf{T}}(t)z(t). \tag{67}$$

The wave variables $u_p(t)$ and $v_{cd}(t)$ can be described by the following expressions:

$$u_p(t) = \frac{1}{\sqrt{2b}}(bs(t) + z(t)); \quad v_{cd}(t) = \frac{1}{\sqrt{2b}}(bs(t) - z(t)); \tag{68}$$

where $b \in \mathbb{R}_0^+$.

On the right hand side of Figure 36, the scattering transformation produces discrete wave variables $u_{\mathsf{pd}}[i]$ and $v_{\mathsf{c}}[i]$, which are related to the corresponding discrete control signal $z_c[i] \in \mathbb{R}^m$ and the system's discrete-time output $s_c[i] \in \mathbb{R}^m$ as follows:

$$\frac{1}{2}(u_{pd}^{\mathsf{T}}[i]u_{pd}[i] - v_c^{\mathsf{T}}[i]v_c[i]) = z_c^{\mathsf{T}}[i]s_c[i] \tag{69}$$

The wave variables $u_{pd}[i]$ and $v_c[i]$ can be described by the following expressions:

$$u_{pd}[i] = \frac{1}{\sqrt{2b}}(bs_c[i] + z_c[i]); \quad v_c = \frac{1}{\sqrt{2b}}(bs_c[i] - z_c[i]); \tag{70}$$

The wave variables $u_{pd}[i]$ and $v_{cd}[i]$ are the delayed versions of the wave variables $u_p[i]$ and $v_c[i]$ respectively such that

$$u_{pd}[i] = u_p[i - p(i)]; \quad v_{cd}[i] = v_c[i - c(i)] \tag{71}$$

in which $p(i), c(i) \in \mathbb{N}_0^+$ are the respective delays at time i as shown in Figure 36 as $Z^{-p(i)}$ and $Z^{-c(i)}$.

5.3.3 Variable Passive Sampler and Variable Passive Hold

The blocks VPS and VPH in Figure 36 denote the variable passive sampler and variable passive hold respectively. These blocks represent a pair of sample-and-hold components that can handle adaptive sampling while at same time preserving passivity. These components perform their tasks based on the sampling intervals provided by the adaptive sampling policy described in Section 5.3.1. VPS performs the task of converting continuous time wave variables to discrete time wave variables while VPH performs the task of converting discrete time wave variables to continuous time wave variables.

The VPS and VPH are designed to satisfy the inequality:

$$\int_0^{t_N} (u_p^{\mathsf{T}}(t)u_p(t) - v_{cd}^{\mathsf{T}}(t)v_{cd}(t))dt \geq \sum_{i=0}^{N-1} T_i(u_p^{\mathsf{T}}[i]u_p[i] - v_{cd}^{\mathsf{T}}[i]v_{cd}[i]) \tag{72}$$

To satisfy (72), VPS can be designed to satisfy the following inequality

$$\int_0^{t_N} u_p^{\mathsf{T}}(t)u_p(t)dt \geq \sum_{i=0}^{N-1} T_i u_p^{\mathsf{T}}[i]u_p[i], \tag{73}$$

while VPH can be designed to satisfy

$$\sum_{i=0}^{N-1} T_i v_{cd}^{\mathsf{T}}[i]v_{cd}[i] \geq \int_0^{t_N} v_{cd}^{\mathsf{T}}(t)v_{cd}(t)dt \tag{74}$$

These inequalities ensure that no energy is generated by the sample and hold devices and thus preserve passivity. Let the jth element of the column vectors $u_p(t)$ and $u_p[i]$ be defined as $u_{p_j}(t)$ and $u_{p_j}[i]$, respectively, where $j = 1, ..., m$ and assume that $u_{pj}(t) = 0$; if t< 0. A design of the VPS that ensures condition (73) is given by

$$u_{p_j}[i] = \frac{1}{\sqrt{T_{i-1}T_i}} \int_{t_{i-1}}^{t_i} u_{p_j}(t)dt, \; \forall \, j \; \in \; \{1, \ldots, m\}. \tag{75}$$

where $u_{p_j}[0] = 0$ and $u_{p_j}(t)$ is the continuous-time wave variable.

In a similar manner, a design of the VPH that satisfies the condition in (74) is given by

$$v_{cd_j}(t) = v_{cd_j}[i], \; t \in [t_i, t_{i+1}). \tag{76}$$

The proofs for both (75) and (76) are provided in Section5.4.

5.3.4 Networked Controller

The networked controller provides control law updates at the request of the system, H_{mp} in order for the plant, H_p to track a desired trajectory. Additionally, for the case of r_c as shown in Figure 36, the networked controller introduces an additional bias to the desired local trajectory. As a result of the introduced bias, y_p the output of the plant, H_p, tracks the desired local trajectory, y_d plus an additional offset, r_c due to the bias.

The networked controller, H_c, is an event-based proportional controller defined as

$$z_c[i] = H_c(f_c[i]) = K_c(s_c[i] - \Lambda r_c[i]) \tag{77}$$

where the proportional gain, $K_c = diag[K_{c1}, K_{c2}, ..., K_{cm}]$ is a positive diagonal matrix. $r_c \in \mathbb{R}^m$ is the bias reference input and Λ is a positive diagonal matrix used in (53).

The networked controller is strictly input passive [99] with an input-output mapping : $f_c \mapsto z_c$. This implies that

$$\sum_{i=0}^{N-1} T_i z_c^{\mathsf{T}}[i] f_c[i] \geq \delta \sum_{i=0}^{N-1} T_i f_c^{\mathsf{T}}[i] f_c[i] - \beta_2 \tag{78}$$

where $\delta, \beta_2 \in \mathbb{R}_0^+$

5.4 Analysis

In this section, we present two main analytical results. First, we show that the proposed PBASC architecture is passive. In addition, under certain assumptions, we show that the overall system, with an input-output mapping $r_c \mapsto s$, is strictly-output passive. Secondly we show that the proposed architecture achieves trajectory tracking of a local reference both in the presence and absence of bias introduced at the networked controller.

5.4.1 Passivity

First, we introduce the following two lemmas to show that the variable sampling-and-hold components, VPS and VPH, are passive by design.

Lemma 5.1. *The proposed VPS given by (75) satisfies the variable passive-sampler condition in (73).*

Proof. Combining (75) and the Cauchy-Schwarz inequality with $u_p[0] = 0$ we have

$$\begin{aligned}
\sum_{i=0}^{N-1} T_i u_p^{\mathsf{T}}[i] u_p[i] &= \sum_{i=0}^{N-1} \sum_{j=1}^{m} T_i u_{p_j}^2[i] \\
&= \sum_{i=1}^{N-1} \sum_{j=1}^{m} T_i \left(\frac{1}{\sqrt{T_i T_{i-1}}} \int_{t_{i-1}}^{t_i} u_{p_j}(t) dt \right)^2 \\
&\leq \sum_{i=1}^{N-1} \sum_{j=1}^{m} \frac{1}{T_{i-1}} \int_{t_{i-1}}^{t_i} u_{p_j}^2(t) dt \int_{t_{i-1}}^{t_i} dt \\
&\leq \int_0^{t_N} u_p^{\mathsf{T}}(t) u_p(t) dt.
\end{aligned}$$

□

Lemma 5.2. *The proposed VPH given by (76) satisfies the variable passive-hold condition in (74).*

Proof. From (76) and with $t_0 = 0$, we have

$$\begin{aligned}
\int_0^{t_N} v_{cd}^{\mathsf{T}}(t) v_{cd}(t) dt &= \sum_{i=0}^{N-1} \sum_{j=1}^{m} v_{cd_j}^2[i] \int_{t_i}^{t_{i+1}} dt \\
&= \sum_{i=0}^{N-1} T_i v_{cd}^{\mathsf{T}}[i] v_{cd}[i]
\end{aligned}$$

□

Next, we present the following theorem that ensures the passivity of the proposed PBASC architecture.

Theorem 5.3. *Consider the proposed PBASC architecture shown in Fig. 36, if the components of the architecture satisfy their individual passivity constraints such that H_{mp} is passive, VPH and VPS are both passive, the networked controller is strictly input passive and the assumptions* ***A5*** *and* ***A6*** *hold, then the closed loop system described in Fig. 36 is passive. Additionally, if $K_c = bI$ and $r_p(t) = \mathbf{0}$, the input-output mapping $r_c \mapsto s$ shown in Fig. 36 is strictly output passive [99].*

Proof. Multiplying both sides of (72) by $\frac{1}{2}$, and substituting (67) we have

$$\int_0^{t_N} (s^{\mathsf{T}}(t) z(t)) dt \geq \frac{1}{2} \sum_{i=0}^{N-1} T_i (u_p^{\mathsf{T}}[i] u_p[i] - v_{cd}^{\mathsf{T}}[i] v_{cd}[i]) \tag{79}$$

If the assumption that no duplicate packet is processed holds and from (71) we can guarantee that

$$\sum_{i=0}^{N-1} T_i(u_p^{\mathsf{T}}[i]u_p[i] - v_{cd}^{\mathsf{T}}[i]v_{cd}[i]) \geq \sum_{i=0}^{N-1} T_i(u_{pd}^{\mathsf{T}}[i]u_{pd}[i] - v_c^{\mathsf{T}}[i]v_c[i]) \tag{80}$$

Multiplying both sides of (80) by $\frac{1}{2}$, and using (69) we have

$$\frac{1}{2}\sum_{i=0}^{N-1} T_i(u_p^{\mathsf{T}}[i]u_p[i] - v_{cd}^{\mathsf{T}}[i]v_{cd}[i]) \geq \sum_{i=0}^{N-1} T_i z_c^{\mathsf{T}}[i]s_c[i] \tag{81}$$

From (79) and (81), we obtain

$$\int_0^{t_N}(s^{\mathsf{T}}(t)z(t))dt \geq \sum_{i=0}^{N-1} T_i z_c^{\mathsf{T}}[i]s_c[i] \tag{82}$$

Substituting $s_c = f_c - r_c$ and $z = r_p - f_p$ in (82) and after further rearrangement and simplification we have

$$\int_0^{t_N}(s^{\mathsf{T}}(t)r_p(t))dt + \sum_{i=0}^{N-1} T_i z_c^{\mathsf{T}}[i]r_c[i] \geq \sum_{i=0}^{N-1} T_i z_c^{\mathsf{T}}[i]f_c[i] + \int_0^{t_N}(s^{\mathsf{T}}(t)f_p(t))dt \tag{83}$$

From (12) and substituting (78) in (83) and with $f_c = {K_c}^{-1}z_c$ we have

$$\int_0^{t_N}(s^{\mathsf{T}}(t)r_p(t))dt + \sum_{i=0}^{N-1} T_i z_c^{\mathsf{T}}[i]r_c[i] \geq \delta\sum_{i=0}^{N-1} T_i z_c^{\mathsf{T}}[i]{K_c}^{-2}z_c[i] - \beta_1 - \beta_2 \tag{84}$$

Let η_{min}>0, be defined as the minimum diagonal element of K_c, then (84) can be rewritten as

$$\int_0^{t_N}(s^{\mathsf{T}}(t)r_p(t))dt + \sum_{i=0}^{N-1} T_i z_c^{\mathsf{T}}[i]r_c[i] \geq \frac{\delta}{\eta_{min}^2}\sum_{i=0}^{N-1} T_i z_c^{\mathsf{T}}[i]z_c[i] - \beta_1 - \beta_2 \tag{85}$$

Thus, the closed loop system is passive.

Next we show that with $r_p = \mathbf{0}$, the input-output mapping $r_c \mapsto s$ is strictly output passive [99][321]. With $r_p = 0$, we have that

$$\int_0^{t_N}(s^{\mathsf{T}}(t)f_p(t))dt = \int_0^{t_N}(s^{\mathsf{T}}(t)(-z(t)))dt \geq -\beta_1 \tag{86}$$

From the expresion of $v_{cd}(t)$ in (68), solving for z(t) and substituting in (86) we have

$$\sqrt{2b}\int_0^{t_N}(s^\mathsf{T}(t)v_{cd}(t))dt \geq b\int_0^{t_N}(s^\mathsf{T}(t)s(t))dt - \beta_1 \tag{87}$$

Substituting $z_c[i] = K_c s_c[i] - K_c\Lambda r_c[i]$ in the expression for v_c defined in (70) and with $bI = K_c$, simplifying we have

$$v_c = \frac{-1}{\sqrt{2b}}K_c\Lambda r_c \tag{88}$$

$\forall t \in [t_i, t_{i+1})$ and based on the expression in (71), substituting $v_{cd}(t)$ in (87) and let $\forall t \in [t_i, t_{i+1})$ $r_{cd}(t) = r_c[i - c(i)]$ and simplifying we have

$$\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}}(s^\mathsf{T}(t)K_c\Lambda r_{cd}(t))dt \geq b\int_0^{t_N}(s^\mathsf{T}(t)s(t))dt - \beta_1 \tag{89}$$

Assuming the maximum diagonal component of $K_c\Lambda = \sigma_{\max}$ then from (89) we have that

$$\int_0^{t_N}(s^\mathsf{T}(t)r_{cd}(t))dt \geq \frac{b}{\sigma_{\max}}\int_0^{t_N}(s^\mathsf{T}(t)s(t))dt - \frac{\beta_1}{\sigma_{\max}} \tag{90}$$

Hence, based on the definition of strictly output passivity in (13), we conclude from (90), that the input-output mapping $r_c \mapsto s$, is strictly output passive. $\square$

5.4.2 Tracking

Theorem 5.4. *Consider the proposed PBASC architecture shown in Fig. 36, with the system, H_{mp} and the event-based networked controller described by (77). Assuming the disturbance input, $r_p(t) = \mathbf{0}$, then*

$$\lim_{t\to\infty} e_a(t) = \lim_{t\to\infty} y_p(t) - y_{da}(t) = \mathbf{0}. \tag{91}$$

where y_{da} and e_a are the modified desired trajectory and modified tracking error respectively in the presence of a bias r_c.

Proof. The system output variable s_a, can be stated as

$$s_a = s - \Lambda r_c = \dot{y}_p - \dot{y}_d + \Lambda y_p - \Lambda y_d - \Lambda r_c$$

Rearranging the terms,

$$s_a = \dot{y_d} - \dot{y}_{da} + \Lambda y_d - \Lambda y_{da} = \dot{e}_a + \Lambda e_a$$

where y_{da}=$y_d + r_c$ is the new adjusted trajectory including the effect of introduced bias and e_a is the adjusted tracking error. Note that the expression for s_a is similar to that described in (53), the only difference being the presence of the bias input, r_c, introduced by the networked controller. Based on Fig. 36, if $f_c = s_{ca} = s_c - \Lambda r_c$, then a corresponding output of the networked controller, z_{ca} be defined as

$$z_{ca} = K_c s_{ca}; \tag{92}$$

Assuming no delays and packet losses, we can define the following:

$$u_{pd}[i] = u_p[i]; \quad v_{cd}[i] = v_c[i]; \tag{93}$$

Also, the wave variables from both the and controller sides of the network described in (68) and (70) respectively using the new variables, s_a, z_a, z_{ca} and s_{ca} can then be concisely described by

$$u_p = \frac{1}{\sqrt{2b}}(bs_a + z_a); \quad u_{pd} = \frac{1}{\sqrt{2b}}(bs_{ca} + z_{ca}); \tag{94a}$$

$$v_{cd} = \frac{1}{\sqrt{2b}}(bs_a - z_a); \quad v_c = \frac{1}{\sqrt{2b}}(bs_{ca} - z_{ca}); \tag{94b}$$

Substitute (94a) into (75) and from (92), solving for s_{ca}, we have

$$s_{ca}[i] = \frac{1}{(b + K_c)} \frac{1}{\sqrt{T_i T_{i-1}}} \int_{t_{i-1}}^{t_i} (bs_a + z_a)(\tau) d\tau \tag{95}$$

Next, substitute (94b) into (76) and from (92) we

$$(bs_a - z_a)(t) = (b - K_c) s_{ca}[i] \tag{96}$$

Substitute (95) in (96)

$$(bs_a - z_a)(t) = \frac{(b - K_c)}{(b + K_c)} \int_{t_{i-1}}^{t_i} (bs_a + z_a)(\tau) d\tau$$

Assuming steady-state conditions, we have that $T_i = T_{i-1} = T$, we have

$$(bs_a - z_a) = \frac{(b - K_c)}{(b + K_c)} (bs_a + z_a)$$

After further simplification, solving for z_a, results in

$$z_a = K_c s_a \tag{99}$$

With the system considered to be passive from $f_{pa} \mapsto s_a$, from (99), $f_{pa} = -z_a = -K_c s_a$. Assuming **A2** holds and with $r_p = \mathbf{0}$, the system is asymptotically stablized at the origin $s_a = 0$, which implies that $\lim_{t \to \infty} e_a(t) = 0$. Based on this, $y_p = y_{da}$, hence guaranteeing tracking of the adjusted trajectory. $\square$

Corollary 5.5. *In the absence of a networked controller bias, that is with the input vector $r_c = \mathbf{0}$, the plant system, H_p tracks the desired trajectory.*

5.5 Simulation Results

We present a case study which involves the trajectory tracking control of a robotic manipulator over a wireless network using the proposed PBASC architecture. First, we provide the dynamic model of the robotic manipulator and then present the derivation of the passive system mapping, H_{mp} and the relationship between the tracking error, e, and the system output, s. Using the derived mapping we present results of the evaluation.

5.5.1 Derivation of H_{mp} and H_{es} for the robotic manipulator

In the absence of friction and disturbances, the Euler-Lagrange equations of motion for an n-degree-of-freedom robotic manipulator can be generally described by [19]:

$$M(y_p)\ddot{y_p} + C(y_p, \dot{y_p})(\dot{y_p}) + g(y_p) = \tau; \tag{100}$$

where $y_p(t) \in \mathbb{R}^n$ is the vector of joint angles, $\tau(t) \in \mathbb{R}^n$ is the input torque vector, $M(y_p) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $C(y_p, \dot{y_p}) \in \mathbb{R}^{n\times n}$ is the matrix of centrifugal and coriolis effects, and $g(y_p) \in \mathbb{R}^n$ is the gravity vector. In order to obtain the passive mapping, H_{mp} and the tracking error function, H_{es} for the robotic manipulator in the form described in Section 5.2, we sought the sliding-mode technique employed in [20] where s is introduced as a sliding variable. The detailed description is as follows:

With the tracking error, e defined as $y_p(t) - y_d(t)$, we choose the local controller, H_{lc} as:

$$\tau = M(y_p)\ddot{y_r} + C(y_p, \dot{y_p})\dot{y_r} + g(y_p) + f_p; \tag{101}$$

where f_p is the input to the system, H_{mp}. Let y_r and $\dot{y_r}$ be defined as

$$y_r = y_d - \lambda \int_0^t e; \quad \dot{y_r} = \dot{y}_d - \Lambda e; \tag{102}$$

where Λ, as defined in Section 5.2, is a positive diagonal matrix. By combining (100), (101) and (102), we obtain the following

$$M(y_p)\dot{s} + C(y_p, \dot{y_p})s = f_p; \tag{103}$$

where s is defined as

$$s = \dot{y_p} - \dot{y_r} = \dot{e} + \Lambda e. \tag{104}$$

The expresion in (103) results in the system, H_{mp}, a passive input-output mapping from $f_p \mapsto s$. Additionally, (104) results in the desired function, H_{es} relating the system output, s to the tracking error, e. Hence, the obtained H_{mp} and H_{es} satisy the assumptions stated in Section 5.2, we can then use the mapping in our PBASC framework for tracking purposes.

5.5.2 Evaluation

We evaluate the robotic manipulator case study using simulations. The setup involves the passive mapping, H_{mp} for the robot and an event-based networked controller, H_c, communicating over a wireless network as shown in Figure 36. The dynamics of H_{mp} and H_c are implemented using Matlab/Simulink models while TrueTime is used to model the wireless network dynamics. The network protocol used is 802.11b, with a speed/bandwidth of 11Mbps. The robot has four degrees-of-freedom and is modeled using four points of mass. The point masses are: $m_1 = 0.362kg$, $m_2 = 0.401kg$, $m_3 = 0.059kg$ and $m_4 = 0.177kg$. The design parameters for the self-triggered policy are $T_{min} = 0.01s$, $T_{max} = 0.1s$, $\Delta = 0.001s$ and $P = 0.5 * I$. The other parameters are wave impedance, $b = 1$, controller gain, $K_c = I$, and the design parameter, $\Lambda = 10 * I$.

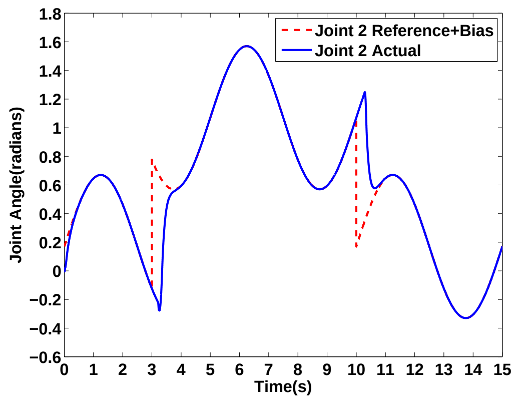
The goal of the approach is for the robot to track a specified trajectory while efficiently using the network resources and maintaining stability. We focus on joint 2 of the robotic arm with a rest position of 0.17 radians. The desired local trajectory with respect to the home position is $y_{d2} = 0.5sin(\frac{2\pi}{5})$.

We also evaluate the introduction of a bias, by the networked controller, which modifies the desired reference trajectory. This bias can be viewed as the presence of an obstacle in the robotic manipulator's path which can only be perceived by the networked controller.

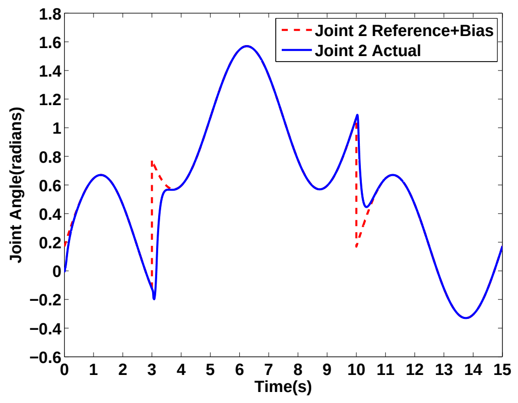
5.5.2.1 PBASC Approach vs Traditional Approach.

This scenario considers nominal network conditions with no additional delays and packet losses. We model the presence of an obstacle as a step reference input, $r_{cj} = 0.9$, which is introduced in the robot's environment between the interval from 3 seconds to 10 seconds. Figures 37a and 37c show the trajectory of joint 2 and the sampling intervals plots respectively using our PBASC approach.

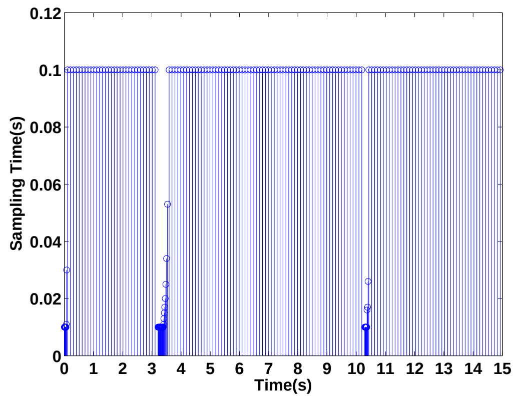
We compare the plots from our nominal scenario with the case of a traditional sampling alternative in the same framework but instead using a fixed sampling period (FSP) of T_{min}. Figures 37b and 37d show the trajectory and sampling intervals plots respectively for the FSP approach. Comparatively, it can be seen that both approaches closely track the specified trajectory and in the presence



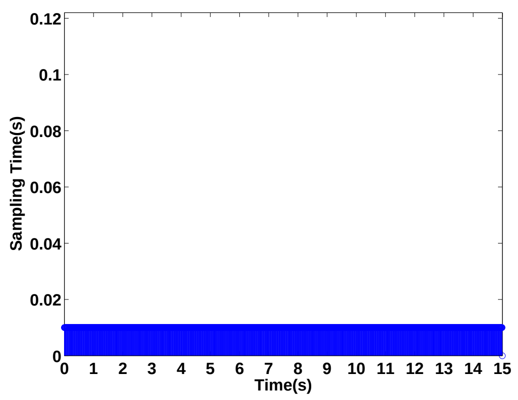
(a) PBASC Trajectory.



(b) FSP Trajectory.



(c) PBASC Sampling Intervals.



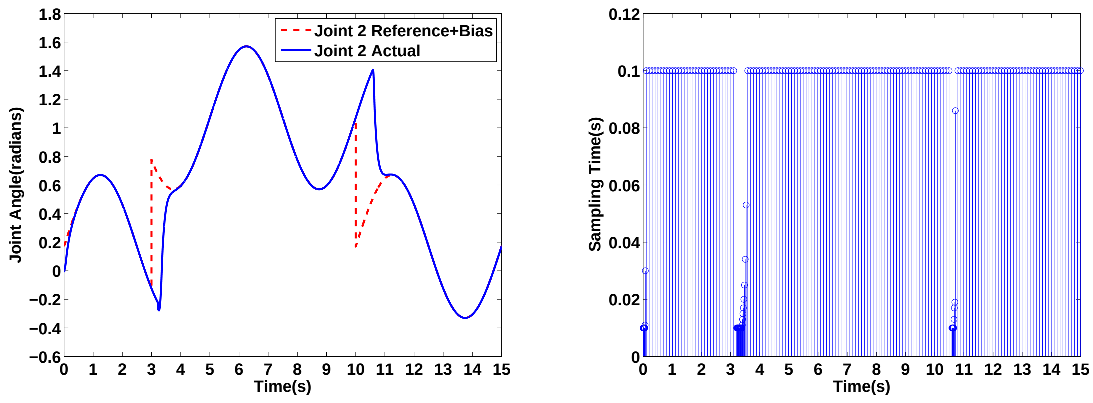
(d) FSP Intervals.

Figure 37: Simulation Results - PBASC Approach vs. FSP Approach in Nominal Case.

an obstacle the trajectories are adjusted with a bias of 0.9 although the FSP approach seems to respond quicker initially. The average tracking error, the average absolute difference between the joint's trajectory and the desired trajectory for the PBASC and FSP approaches are 0.1064 and 0.04 respectively. On the other hand, using PBASC approach, fewer control law updates are required in order to maintain tracking thereby reducing the amount of network resources utilized.

5.5.2.2 Time-Varying Delays.

This scenario considers the effect of time-varying delays. To simulate the case of time-varying delays, starting the nominal case we incorporate a disturbance node in the network with a sampling period of 0.05 seconds. The disturbance node floods the network with packets based on a Bernoulli process with parameter d. The disturbance node samples a uniformly distributed random variable X[k]$\in [0, 1]$ every 0.05 seconds. If X[k] $>$ d, a disturbance packet is forced on the network.



(a) PBASC Trajectory(Delay). (b) Sampling Intervals (Delay).

Figure 38: Simulation Results - PBASC Approach with Introduced Delays.

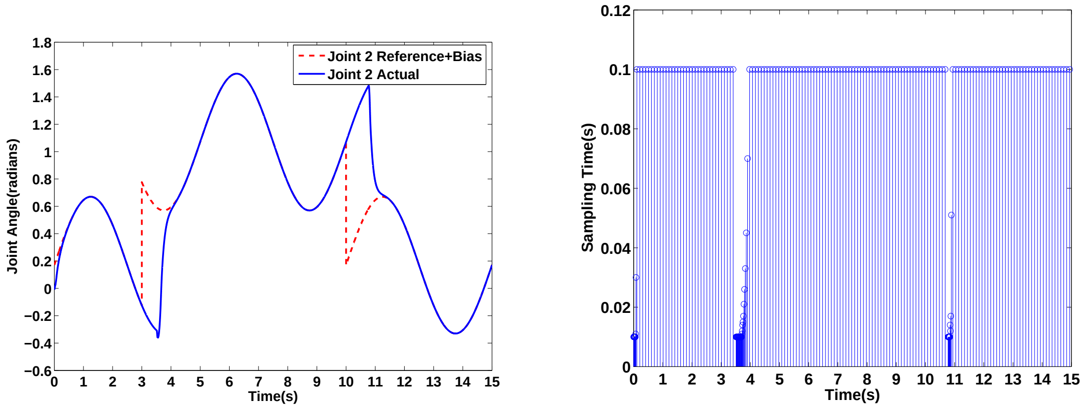
Figure 38a shows the trajectory of joint 2 in the presence of time-varying delays. Compared to the nominal case using the PBASC approach, due to the presence of time delays it takes the system a little longer to adjust it's trajectory in order to track the modified trajectory. The average tracking error in the presence of time delays is 0.117. Also, the impact of the delay can also be seen in Figure 38b, as more control updates are requested in order to achieve tracking. The overall system still maintains stability in the presence of time-varying delays.

5.5.2.3 Packet Losses.

This scenario demonstrates the effect of packet losses on the PBASC approach. We consider a lossy network whereby packets containing either sensor updates or control signals can be lost in the communication channel. The packet loss conditions are designed using a probabilistic Bernoulli loss model. A packet is dropped if a sampled probability is less than a specified probability. We consider the case of thirty percent probability of packet loss. The average tracking error with time delays is 0.1485. From Figures. 39a and 39b, the effect of packet losses is clearly evident from the plots by the deteriorated performance in tracking but the overall system remains stable.

5.6 Experimental Results

In this section, we present an experimental evaluation of the proposed PBASC using the networked robotic manipulator case study described in Section 5.5. In this evaluation, instead of the using a model of the robot and TrueTime for modeling the network and platform as in the Section 5.5,



(a) PBASC Trajectory(Loss).   (b) Sampling Intervals (Loss).

Figure 39: Simulation Results - PBASC Approach with Packet Loss.

we use an actual robotic manipulator and an actual wired/wireless network. We will first present the experimental setup for the evaluation and then we provide a brief overview of the Real-Time Windows Target (RTWT) software. Subsequently, we present the various scenarios considered and the results from each scenario.

5.6.1 Experimental Setup

The setup for the experimental evaluation of the proposed PBASC is depicted in Figure 40. On the right-hand side of Figure 40, the robotic manipulator is connected to a computing unit, denoted C1, through a USB2Dynamixel controller. The computing unit, C1, running RTWT, executes the software components on the plant side of the network which includes the local controller, variable passive sampler and variable passive hold, the adaptive sampling scheme, wave transformation, the local reference and the robot's interface software. The computing unit, C2, on the left-hand side of the network executes the event-based networked controller, wave transformation and reference input. The communication between the plant and controller is over an wired/wireless network.

5.6.2 Real-Time Windows Target

Real-Time Windows Target (RTWT) is a Matlab/Simulink rapid prototyping software which provides a PC solution for designing and executing real-time systems. RTWT allows one to use a PC as a standalone self-hosted target for running Simulink models interactively in real-time. RTWT supports direct I/O, providing real-time interaction with Simulink models, making it an easy-to use,

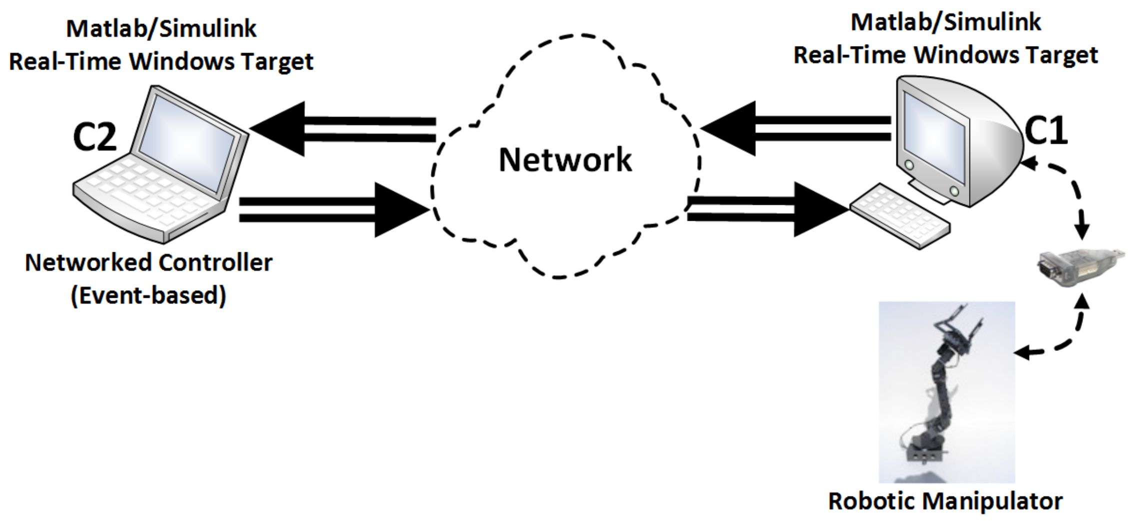


Figure 40: Experimental Platform Configuration

low-cost target environment for rapid prototyping and hardware-in-the-loop simulation. RTWT can be operated in two main modes, *normal mode* and *external mode*. In the *normal mode*, one can use the standard Simulink environment together with components to create and simulate designed models. In order to execute the model in real-time, RTW and an installed C/C++ compiler is used to generate executable code which can then be run in real time with Simulink in *external mode*. The built-in functionality of RTW compiles the Simulink model down to C code, and then builds a native executable file [322, 323, 324].

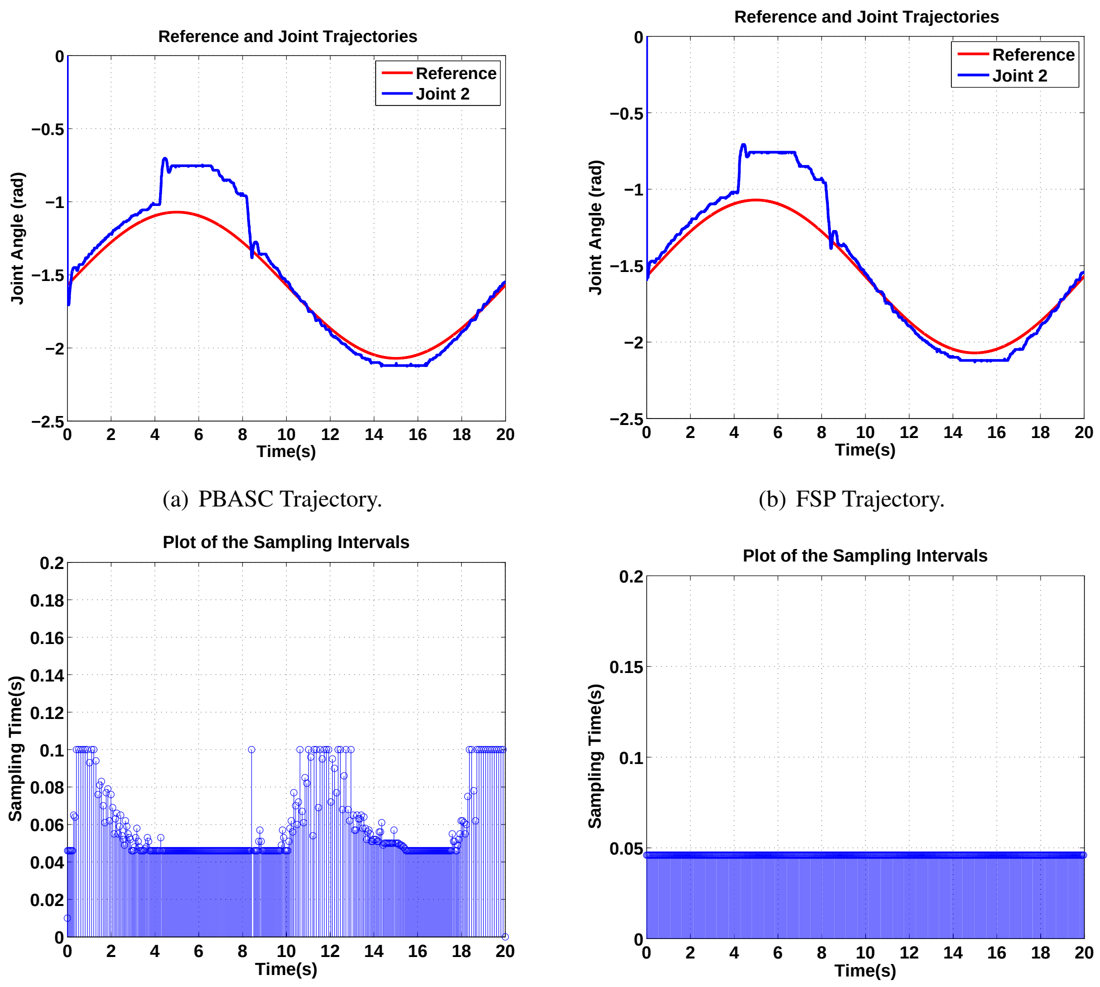
RTWT uses a small real-time kernel to ensure deterministic sampling rate in the application. During operation, the real-time kernel runs at CPU ring zero (privileged or kernel mode) and uses the PC clock as its primary source of time. The scheduler in RTWT allows one to work with a single sample rate or with multiple and potentially different sampling rates in a model. In RTWT, each sampling rate is defined as a task and is clocked by the scheduler that runs the executable. RTWT supports up to a maximum of 32 tasks, and faster tasks have higher priority. For non-real time simulation which is done in the *normal mode*, Simulink uses a computed vector to step a designed model. On the completion of output computation for a given time value, computations for the next time value is repeated. This process continues till the stop time is reached. Since the computed time vector is not connected to a hardware clock the computation is done in non-real time and typically performed as fast as the computer can run. On the other hand, for real-time execution, which is done in *external mode*, RTWT uses interrupts to step the designed application in real time at the proper rate [322, 323].

The experimental evaluations for PBASC were performed using RTWT's real-time execution in *external mode*. The serial communication options in the Packet-In and Packet-Out blocks from the RTWT library were used to locally interface with the robotic manipulator via the dynamixel controller. For the communication between C1 and C2, through which the remote plant and controller exchange wave variables, the UDP options of the Packet-In and Packet-Out blocks were utilized.

5.6.3 Results

5.6.3.1 Experiment 1: Nominal Case

In this experiment, we consider nominal network condition whereby the remote plant and the networked controller exchange information over 100 BASE-TX Ethernet network without any additionally introduced network uncertainties. For the nominal case we perform two experiments, one based on a fixed sampling interval, denoted FSP, and the other based on our proposed PBASC framework. The experiments are performed using the setup as described in Section 5.6.1. Figures 41a and 41b show the reference and actual trajectories of joint 2 of the robotic manipulator for both the PBASC and FSP approaches while Figures 41c and 41 show the sampling interval for each corresponding approach. A reference input or bias of 0.2 rad is introduced at the networked controller between 5s and 9s. This can be clearly see from the trajectories for both approaches and after the bias is removed the joint again starts to follow the local reference trajectory. Based on the plots, the robotic manipulator was able to track the reference trajectories for both the FSP and PBASC approaches with a little noticeable error. From the sampling interval plots, one can clearly observe the benefits of the PBASC approach, while the FSP requests controller updates at a fixed sampling interval of 46ms, PBASC's controller update requests are based on the tracking error. For the 20s experiment, the FSP requested a total of 434 controller updates compared to PBASC's 329 controller update requests.



(a) PBASC Trajectory.

(b) FSP Trajectory.

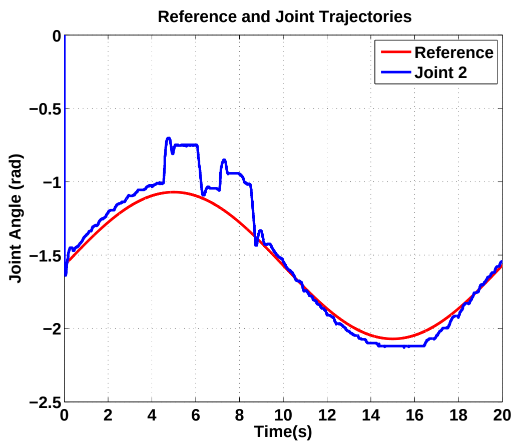
(c) PBASC Sampling Intervals.

(d) FSP Intervals.

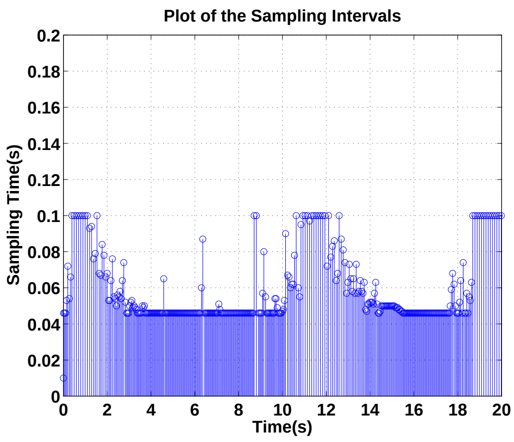
Figure 41: Experimental Results - PBASC Approach vs. FSP Approach in Nominal Case.

5.6.3.2 Experiment 2: Persistent Link Interruption

In this experiment, we illustrate, the robustness of the proposed PBASC framework to network uncertainties. We introduced a Boolean variable in order to artificially model link disruption or interruption by toggling the variable. During the time interval from 6s to 7s, we interrupted the link from the controller to the plant using the toggle block. From Figure 42a, one can observe that the system still maintains stability, the joint resorts intermittently to tracking the local reference and when the link is reestablished the joint goes back to tracking the modified reference trajectory with the introduced networked reference. Figure 42 shows the sampling interval clearly showing increased sampling during the intervals of larger error.



(a) PBASC Trajectory(Persistent Link Interruption).

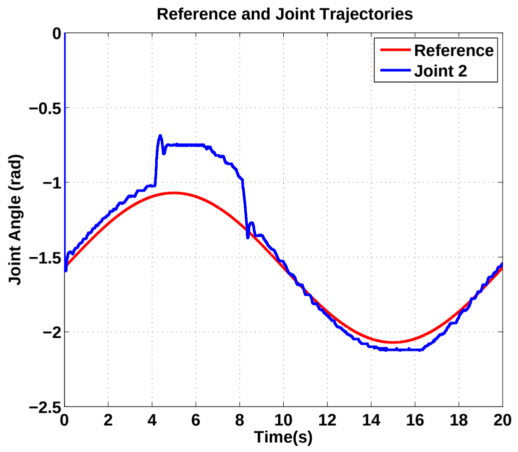


(b) Sampling Intervals (Persistent Link Interruption).

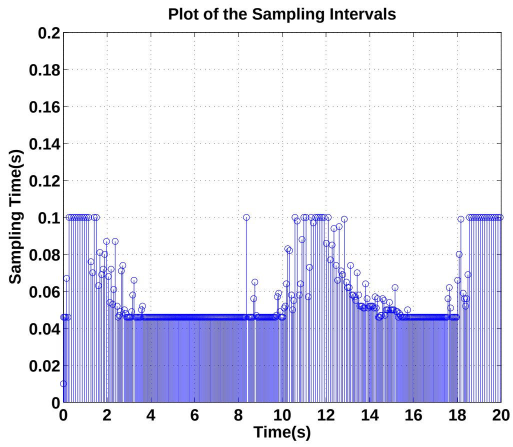
Figure 42: Experimental Results - PBASC Approach with Link Interruption.

5.6.3.3 Experiment 3: Intermittent Wireless Connection

In this experiment, we evaluate the impact of unreliable wireless network on the proposed approach. For this experiment, instead of the wired network used in the two previous scenarios, we switched to wireless communication. Specifically, we disconnected the wired communication for C2, the computer running the networked controller, and connected to the Ethernet network through an 802.11b wireless connection for information exchange with, C1, the computing unit running the remote robot. Figure 43a and 43b shows the joint 2 trajectory and sampling intervals respectively. Even with the wireless communication the robot still achieves tracking with reduced sampling interval compared to the FSP approach.



(a) PBASC Trajectory(Intermittent Wireless Connection).



(b) Sampling Intervals (Intermittent Wireless Connection).

Figure 43: Experimental Results - PBASC Approach with Intermittent Wireless Connection.

5.7 Summary

In this chapter, an integrated passivity-based adaptive sampling control architecture for trajectory tracking is introduced. The approach integrates passivity and adaptive sampling to address the challenges of guaranteeing stability in the presence of network uncertainties and efficiently utilizing limited network resources while at the same time achieve a trajectory tracking objective. Simulations and experimental results are presented for a case study involving the trajectory tracking control of a robotic manipulator over a network in the presence of network uncertainties. Compared to the fixed sampling approach, the adaptive sampling framework utilizes the network resources more efficiently which is very desirable especially in the case of limited resources.

CHAPTER 6

INTEGRATED MODELING AND SIMULATION OF NETWORKED CONTROL SYSTEMS

6.1 Introduction

NCS are typically designed based on simplifying assumptions on the network, specifically in regards to their network operating environment (e.g. time-varying delays and packet losses). These assumptions, although they make the analysis of NCS tractable, often do not fully represent the real network dynamics. Such limitations can lead to catastrophic consequences when the actual systems are deployed, as the overall system behavior depends on network dynamics and uncertainties. As described in Chapter 2, numerous techniques aim to formally analyze NCS properties such as stability, performance, safety, and security but as NCS become increasingly complex, it becomes more challenging to formally analyze these properties. As a result, there is a pressing need to evaluate both the control system and the networking system together for a rapidly growing number of applications, such as unmanned aerial vehicles (UAVs) and industrial control systems. Simulation is a powerful technique for evaluation and can be used at various design stages, but it requires the support of appropriate tools during the design-time and run-time stages in order for the process to be efficient and less prone to errors.

Currently, several simulators have been used for simulating NCS but have limited capabilities. For example, Matlab/Simulink is a very popular tool to model and evaluate the performance of control systems [173]. Although network simulation is provided in Matlab/Simulink using add-ons such as TrueTime [325], the accuracy of the simulation depends on the level of abstraction of the network protocol models. Specifically, the network protocol in TrueTime only supports link layer protocols but not higher level protocols such as TCP or UDP protocols, which are essential for simulating the communication network of a NCS. Packet-level network simulators such as ns-2 [170], provide a detailed implementation of the network stack for packet level data transmission. Yet, using ns-2 only for NCS evaluation requires the control algorithm to be fully implemented in a high-level language such as C++. This becomes very difficult as the complexity of the NCS increases. In order to develop a realistic and accurate simulation of NCS, we need a modeling and simulation environment that can integrate existing tools for the accurate simulation of the control dynamics as well as the networking system of a NCS. The challenge faced by these tools is the ability to precisely model the both dynamical system and the network stack in order to simulate NCS.

The integration of existing tools for the accurate simulation of NCS, although very beneficial, faces several challenges. The first challenge is the design-time scalability of modeling NCS which involves the ability to rapidly design and model NCS of various complexity and size. The second challenge is time synchronization of the heterogeneous simulation components during execution. Given that the simulators operate in potentially different time scales using disparate time models, time synchronization between the simulators is critical to preserve the correctness of the simulation. The third challenge involves the data communication between the simulators to ensure consistent data semantics during the simulation. Finally, the fourth challenge involves the run-time scalability which is the ability of the simulation environment to handle the simulation of large and complex NCS.

In this chapter, in order to address these challenges, we present an integrated modeling and simulation tool for NCS, called the Networked Control Systems Wind Tunnel (NCSWT), which combines the network simulation capabilities of ns-2 with the control design and simulation capabilities of Matlab/Simulink. NCSWT addresses the challenge of design-time scalability of modeling NCS described previously by adopting Model Integrated Computing (MIC) [326]. MIC is an approach for the development of complex software systems, applicable in all phases - analysis, design, implementation, testing, maintenance and evolution. The key idea in MIC is to create domain-specific modeling languages (DSMLs) using a meta-modeling framework and then describe objects in terms of the domain-specific models. These models are formally represented and can be checked for correctness against pre-specified design rules and can be programmatically traversed and transformed to produce/or modify code and other engineering artifacts. Often, these models are transformed into alternate but equivalent representations, which can be used by external analysis and simulation tools to verify certain properties of the system [326]. We present three domain specific modeling languages (DSMLs), the NCSWT model integration language (NCSWT MIL), the Control Design Modeling Language (CDML) and the Network Design Modeling Language (NDML). The DSMLs, developed using the generic modeling environment (GME) [310], facilitate the rapid design and modeling of NCS. In addition, the DSMLs and the NCSWT framework are designed to ensure the consistency of data semantics among the simulators used in the simulation of a NCS.

NCSWT addresses the challenges involving time synchronization and data communication by adopting the High Level Architecture (HLA) for the implementation of the simulation environment framework [184]. HLA is a standard for simulation interoperability that allows independently developed simulations, each designed for a particular problem domain, to be combined into a larger and more complex simulation. In HLA, the independent simulators are known as federates and the larger simulation formed by the interconnection of the federates is known as the federation. The HLA standard provides a set of services to accurately handle time management and data distribution among the heterogeneous simulators. In our framework, we utilize the time management services provided by the HLA to ensure that the time model in the control system simulated in Matlab/Simulink and the time model in the networking system simulated in ns-2 are synchronized. We also utilize the data distribution services to ensure the correct exchange of data between the simulations of the control dynamics and communication network of a NCS.

Additionally, we provide case studies to evaluate the tool. The first case study involves an unmanned aerial vehicle which is remotely controlled over an 802.11b wireless network to follow a given reference trajectory. This case study is used to demonstrate the use of the NCSWT to evaluate the impact of network effects, such as packet loss, time-varying delays and multi-hop topologies, on a safety critical system. The second NCS case study is an industrial control system involving a network of three plant and controller pairs sharing a wireless communication channel and operating at different sampling rates. This case study is used to demonstrate the convenience and scalability of NCSWT tool in handling NCS of different complexities in time and size.

In Chapter 2, we described several efforts have been made toward integrating multiple simulators in order to effectively simulate NCS. In contrast to these efforts, our proposed approach provides a model-based framework that tightly integrates the design of the control system and communication network in NCS providing a well-defined abstraction of the information exchange between the two design views. Also, our integration approach uses a standard based on the HLA for the integration of heterogeneous simulators which assures that the integration of our tools enforces the HLA time synchronization and data communication standards. The rest of the chapter is organized as follows. Section 6.2 provides an overview of the NCSWT. Section 6.3 presents the model-based design and

Table 4: Acronyms

NCS	Networked Control Systems
HLA	High Level Architecture
MIC	Model Integrated Computing
DSML	Domain Specific Modelling language
NCSWT	Networked Control Systems WindTunnel
MIL	Model Integration Language
CDML	Control Design Modelling language
NDML	Network Design Modelling language
TAG	Time Advance Grant
TAR	Time Advance Request
NER	Next Event Request

integration of NCS using the NCSWT tool. Section 6.4 describes the NCSWT run-time components. Section 6.5 presents the implementation overview of the NCSWT tool. Section 6.6 presents the cases studies. Section 6.7 provides an evaluation of the NCSWT and finally Section 6.8 provides a summary of the chapter. For ease of readability, a list of the main acronyms used throughout this chapter is provided in Table 4.

6.2 Overview of the NCSWT

An overview of the NCSWT architecture is shown in Figure 44. The architecture is composed of two main parts, the design-time models and the run-time components. The design-time models are used to define the NCS and its components in order to enable the realization of a HLA-based simulation of the NCS. The design-time models are defined by three domain specific modeling languages (DSMLs), the NCSWT Model Integration Language (NCSWT MIL), the Control Design Modeling Language (CDML) and the Network Design Modeling Language (NDML). These DSMLs, developed using the MIC approach [326], use abstractions to define the NCS in specific modeling domains to enable the rapid modeling and design of NCS for simulation. The design-time models constructed from these DSMLs generate software components, interface glue code and configuration files for the NCS which is deployed and executed at run-time. The run-time components represent the main software components and interfaces for the actual simulation of the NCS using the HLA framework. These components include the Run-Time Infrastructure (RTI), the federates, the simulators (Matlab/Simulink and ns-2) and all the necessary configuration and interface scripts generated from the design-time models. These components also include the monitoring tools for

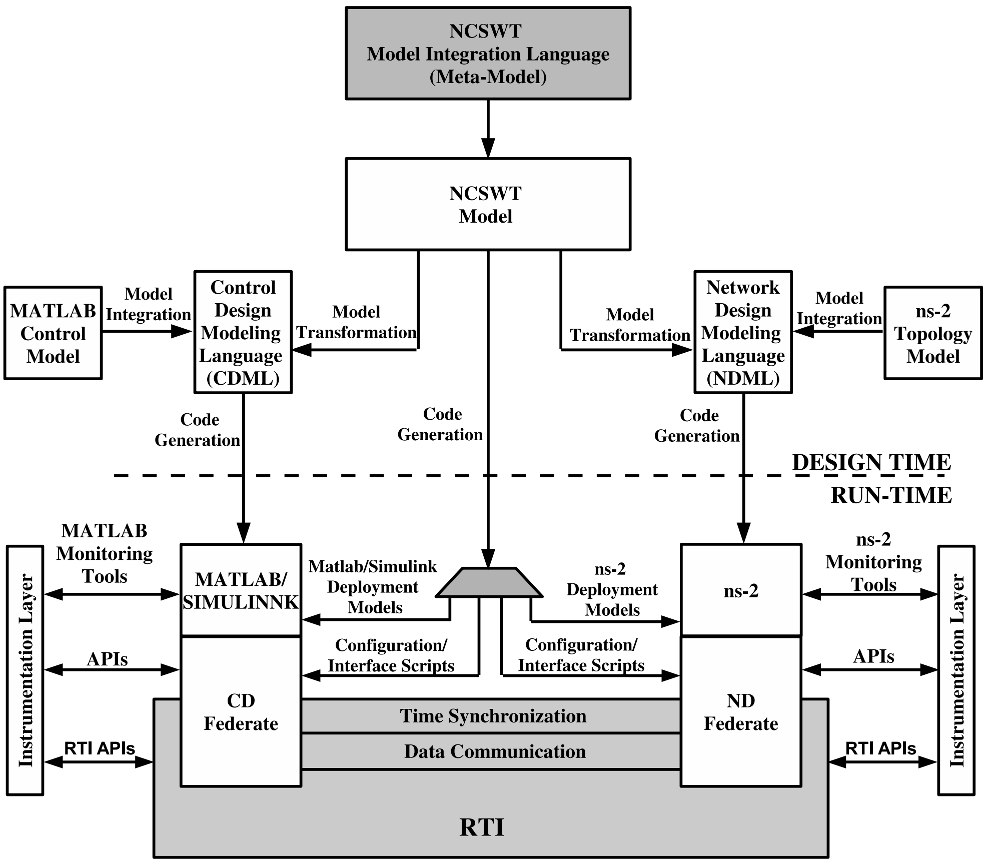


Figure 44: Overview of NCSWT

visualizing and evaluating the results.

A detailed description of the design-time modeling framework and the run-time execution framework are presented in Section 6.3 and Section 6.4 respectively.

6.3 Model-Based Design and Integration

In order to facilitate the rapid generation and simulation of NCS with minimal effort, we employ Model Integrated Computing (MIC) techniques [326]. We define meta-models for three domain-specific modeling languages (DSMLs) for modeling and integrating the NCS in the HLA framework. The DSMLs are:

1. NCSWT Modeling Integration Language (NCSWT MIL)
2. Control Design Modeling Language (CDML)
3. Network Design Modeling Language (NDML)

NCSWT MIL is an extension of the Base HLA Meta-Language [186]. Before we provide a description of each of the DSMLs, we provide a background of the Base HLA Meta-Language.

6.3.1 Base HLA Meta-Language

The Base HLA Meta-Language is a DSML which provides a graphical environment for designing and deploying heterogeneous simulations using model-based design techniques [186]. This DSML provides all of the modeling primitives required to specify the integration, deployment, and execution of a federated simulation. Once the integration model has been defined for a given environment, a set of reusable model interpreters are executed to automatically generate engine-specific glue code and all deployment and execution artifacts. All generation and deployment steps directly rely upon the initial integration model. Figure 45 shows the primary portion of the Base HLA Meta-Language

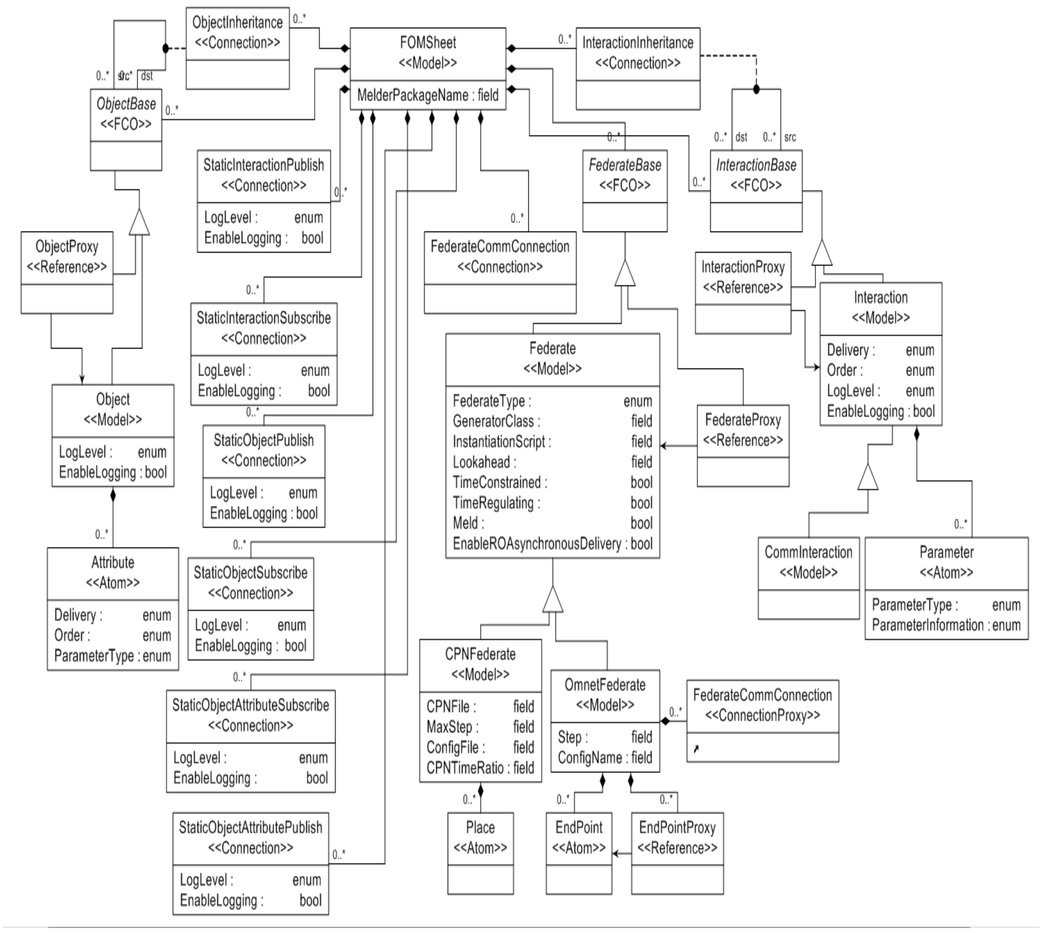

ObjectInheritance <<Connection>>
FOMSheet <<Model>>
MelderPackageName : field
InteractionInheritance <<Connection>>
ObjectBase <<FCO>>
src dst
StaticInteractionPublish <<Connection>> LogLevel : enum EnableLogging : bool
FederateBase <<FCO>>
InteractionBase <<FCO>>
ObjectProxy <<Reference>>
FederateCommConnection <<Connection>>
InteractionProxy <<Reference>>
StaticInteractionSubscribe <<Connection>> LogLevel : enum EnableLogging : bool
Interaction <<Model>> Delivery : enum Order : enum LogLevel : enum EnableLogging : bool
Federate <<Model>> FederateType : enum GeneratorClass : field InstantiationScript : field Lookahead : field TimeConstrained : bool TimeRegulating : bool Meld : bool EnableROAsynchronousDelivery : bool
Object <<Model>> LogLevel : enum EnableLogging : bool
StaticObjectPublish <<Connection>> LogLevel : enum EnableLogging : bool
FederateProxy <<Reference>>
Attribute <<Atom>> Delivery : enum Order : enum ParameterType : enum
StaticObjectSubscribe <<Connection>> LogLevel : enum EnableLogging : bool
CommInteraction <<Model>>
Parameter <<Atom>> ParameterType : enum ParameterInformation : enum
CPNFederate <<Model>> CPNFile : field MaxStep : field ConfigFile : field CPNTimeRatio : field
OmnetFederate <<Model>> Step : field ConfigName : field
FederateCommConnection <<ConnectionProxy>>
StaticObjectAttributeSubscribe <<Connection>> LogLevel : enum EnableLogging : bool
StaticObjectAttributePublish <<Connection>> LogLevel : enum EnableLogging : bool
Place <<Atom>>
EndPoint <<Atom>>
EndPointProxy <<Reference>>
0..*

Figure 45: Base HLA Meta-Language

DSML, specified using GME, that defines the composition elements. The three primary elements in a federation (defined by the model FOMSheet) are Interaction (on right-hand side of Figure 45), Object (on the left-hand side), and Federate (in the center), representing a HLA-interaction, HLA-object, and a HLA-federate respectively. The proxy elements are simply references to their respective target model elements and are used in place of their targets to simplify the presentation of the model. The Federate element directly corresponds to any single instance of a simulation tool involved in the federation. As defined by the HLA standard [184], federates in a federation communicate among each other using HLA-interactions and HLA-objects, both of which are managed by the RTI. Interactions and objects, in an analogy with interprocess communications in operating systems, correspond to message passing and shared memory respectively.

In Figure 45, the ParameterType attribute on the Parameter and Attribute elements defines the data type of that element (i.e. float, int, string). The Interaction and Attribute elements also support the HLA-defined attributes of Delivery and Order [184]. The primary attribute of a federate, as far as HLA-based synchronization is concerned, is its Lookahead, the interval into the future during which that federate guarantees that it will not send an interaction or update an object. The Federate as shown in Figure 45 can correspond to various types such as Java, C++, Matlab/Simulink etc. The language elements StaticInteractionPublish, StaticInteractionSubscribe, StaticObjectPublish, and StaticObjectSubscribe, represent primitives necessary to model federates publishing and subscribing interactions, objects, and attributes [186]. A detailed description can be found in [186][327][328]. With these elements a designer is able to completely specify the integration model of the entire federation and its constituent simulation engines.

6.3.2 NCSWT Model-Based Approach

The NCSWT MIL is an extension of Base HLA Meta-Language. The major extensions are the following:

1. We introduced a new federate to model the ns-2 simulator in the HLA framework and we extended the existing Matlab federate in the Base HLA Meta-Language to specifically capture the design of a NCS.

2. We also introduced new interactions to specifically describe the exchange of information in

NCS.

3. We developed two additional DSMLs that refine the NCS base model from NCSWT MIL by providing specific modeling concepts to model the dynamics of the control design and network design of the NCS.

4. We also provide model interpreters that automatically generate deployable system models for the control design and network design as well as glue code for run-time execution from the DSMLs.

Figure 46 shows the model-based design architecture for NCSWT. The figure shows the design-time models used to define the NCS and its components using representative models in order to enable the simulation of a NCS. Figure 46 shows the three DSMLs, NCSWT MIL, CDML and NDML utilized in our framework. The block "MATLAB control model" represents the Matlab/Simulink control models which are provided as parameters in the CDML. Likewise, the block "ns-2 Topology model" represents the network topology of the NCS provided as parameters in the NDML. From the three DSMLs, a set of reusable model interpreters are used to generate system models, configuration files, deployment models and interface scripts for the simulation of NCS. We provide a description of the DSMLs, and we use an example NCS to define a model for each DSML.

6.3.2.1 NCSWT Model Integration Language

The NCSWT Model Integration Language (NCSWT MIL) provides the modeling primitives to specify the NCS in terms of elements in the HLA framework, such as federates representing the simulators, interactions representing the communication between the simulators, and parameters exchanged in the interactions, in order to execute a HLA-based simulation. An instance model created using NCSWT MIL is referred to as the base architecture of the NCS. The base architecture defines the component of the NCS in terms of federates, interactions and the parameters. The executables for configuring the run-time environment for a specific NCS are generated from the base architecture model. The NCSWT MIL describes the tight coupling between the control and network design views of the NCS by defining how the two design views interact. This tight coupling ensures the consistency of the data semantics among the design views. Figure 47 shows the NCSWT MIL meta-model.

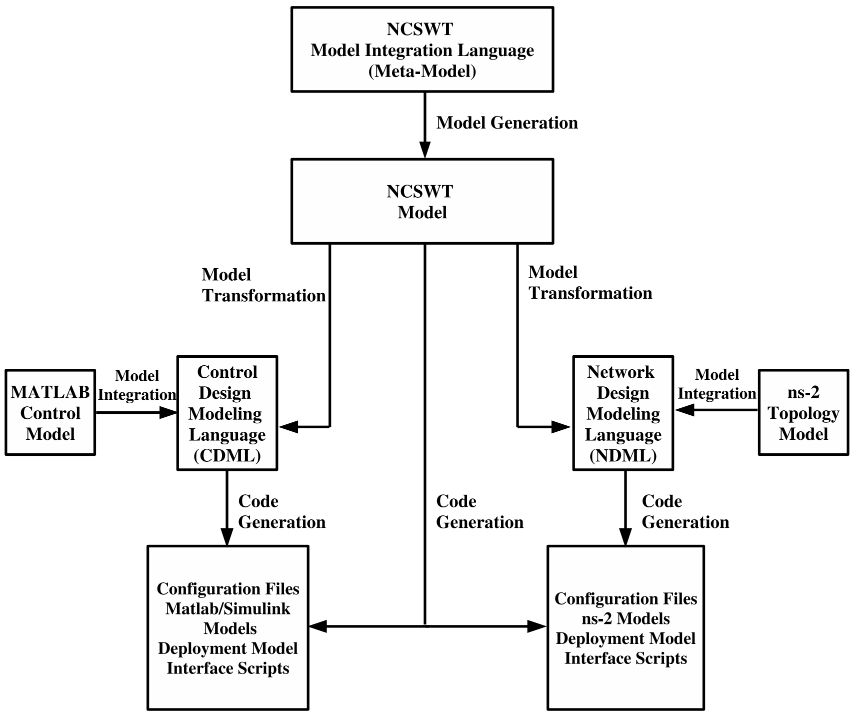


Figure 46: Model-Based Design Architecture for NCSWT

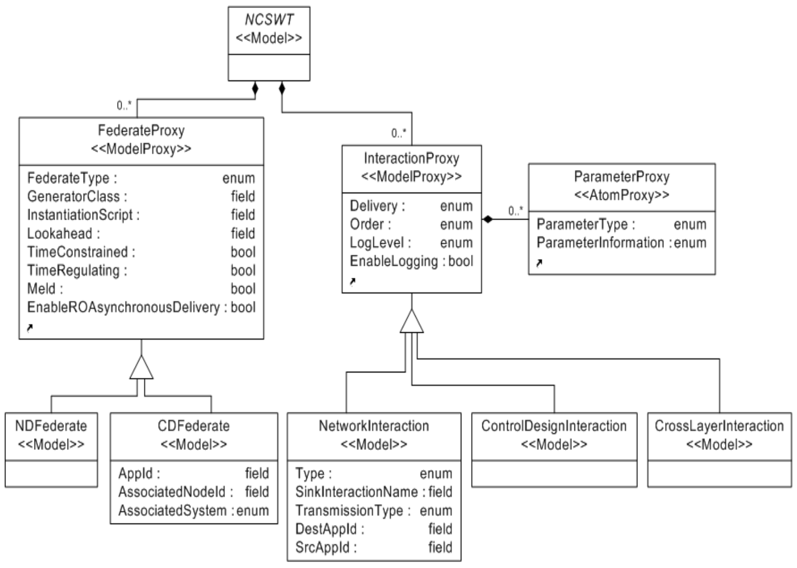


Figure 47: NCSWT Meta Model

1. **Federates:** The federates shown in the NCSWT MIL meta-model represent the elements for describing the simulators used in the NCS simulation based on the HLA framework. The NCSWT MIL models two main types of federates: the ND federate and the CD federate.

   (a) **ND Federate:** This federate models the software component for interfacing the network layer simulator, ns-2, with the RTI.

(b) **CD Federate:** This federate models the software component for interfacing the control layer simulator, Matlab/Simulink, with the RTI. The CD Federate extends the previously-existing Matlab federate in the Base HLA Meta-language [327] with additional attributes. The additional attributes AppID, AssociatedNodeId and AssociatedSystem are used in the definition of parameters in the control and network layers of the NCS.

2. **Interactions:** In the HLA framework, the information exchanged between federates are defined using interactions. In the NCSWT MIL, we introduce three main types of interactions in order to capture the main type of information exchange that can exist in a NCS. The three types of interactions that can be modeled in NCSWT MIL are Network interaction, Crosslayer interaction, and Control Design interaction.

   (a) **Network Interaction:** This interaction type enables the modeling of packets or information exchanged between control design components over the communication network. This includes, for example, the exchange of information between a plant and controller through the communication network such as sending plant outputs to the controller and sending control signals to the plant. In the NCSWT MIL, this interaction type always occurs as a pair with one interaction acting as the source and the other acting as the sink. The source interaction represents communication from a CD federate to the ND federate while the sink interaction represents the communication from the ND federate to another CD federate, which is the destination. This implies that a source network interaction must have a corresponding sink interaction and vice versa.

   (b) **Crosslayer Interaction:** This interaction type allows for the modeling of information exchange between the network and application layers of a network protocol stack. It models the local communication between control design components and their respective local network interface. A typical example of crosslayer interaction is when a control design component queries its network interface for current network condition such as bit rate or loss rate etc. Such information can be used by the control design component to evaluate the quality of service of the network in order to implement a desirable control law or for other objectives.

(c) **Control Design Interaction:** This interaction type allows the modeling of the information exchanged between control design components that are transmitted or received by means other than the communication network. A typical example where this type of interaction is needed is modeling a radar sensor on a UAV for detecting the proximity of an obstacle or another UAV in its vicinity. This information is typically processed at the application (control design) level without being sent over the communication network.

Example

We introduce a NCS example to illustrate the NCSWT MIL. Figure 48 shows a linear-time invariant continuous plant controlled by a proportional derivative (PD) digital controller over a 802.11b wireless network. The objective of the NCS is for the plant to track a reference velocity profile based on feedback information from the controller. The plant output is periodically sent to the digital controller while the control law is periodically sent to the plant from the digital controller. Figure 49

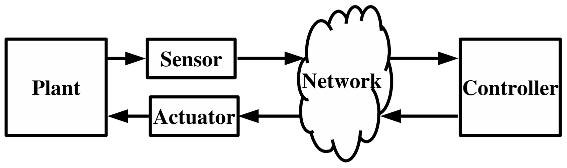


Figure 48: Example Networked Control System

shows a model of the NCSWT MIL for the example scenario. The Plant and the Controller are modeled as CD federates each representing an instance of the Matlab/Simulink simulation tool for each component. The communication network is modeled as a ND federate representing the ns-2 simulator. The blocks PlantIn, PlantOut, ContIn and ContOut are network interactions representing the information exchange between the plant and the digital controller over the network. In this example, the PlantOut, a source interaction, represents sensor information from the Plant, and ContIn is the sink interaction that eventually receives the sensor information after it goes through the network. Similary, the interaction ContOut is the source interaction representing the control signal from the Controller and PlantIn is the corresponding sink interaction that eventually receives the control signal after it goes through the network.

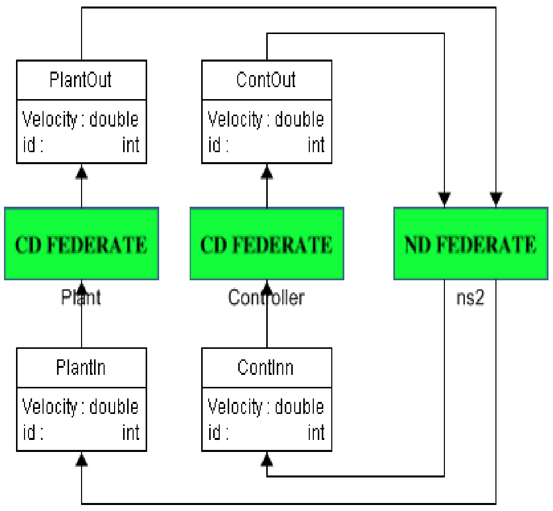


Figure 49: NCSWT Model

6.3.2.2 Control Design Modeling Language

The Control Design Modeling Language (CDML) defines the modeling elements for describing the dynamic behavior of the system components of the NCS. Figure 50 shows the metamodel of the CDML, which defines control design concepts representing the control design view of a modeled NCS. The main components are the system type and the type of connection between system model components in the NCS. The system type models the dynamical behavior of the components of the

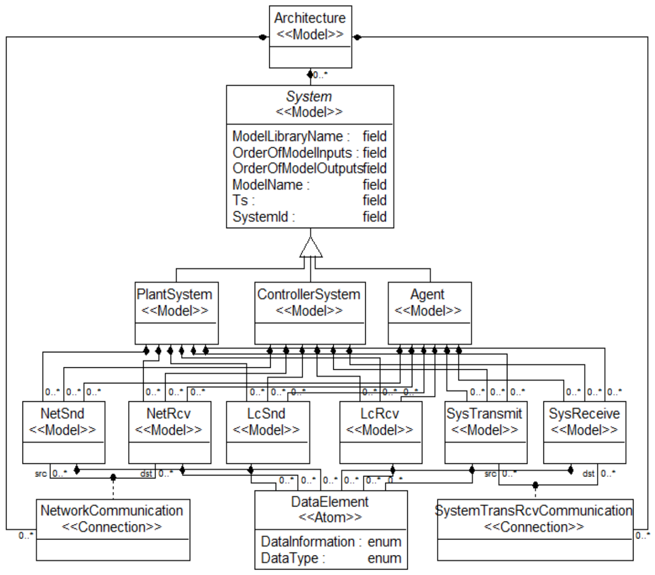


Figure 50: Control Design Meta Model

NCS and can be modeled as one of three types.

1. **Plant:** This models the dynamics of the system to be controlled.

2. **Controller:** This models the control law or algorithm to modify the behavior of the plant to behave in a desired manner.

3. **Agent:** This essentially models a dynamic component which can neither be categorized specifically as a plant or a controller. For example, an UAV in a network of UAVs whose individual behavior is controlled by its neighbors.

A model created from the CDML is a refinement of the base architecture model of a NCS, created by the NCSWT MIL, with the details regarding the dynamics of the control system defined. In order to ensure consistency with the base architecture model defined in the NCSWT ML, a model transformation is used to transform a base architecture model directly to a base model in CDML. Then the control design concepts in CDML are used to define the dynamics of the components in the NCS.

It should be noted that CDML does not re-implement all of the Matlab/Simulink syntax and blocks, it provides an interface for generating and building specified control design models based on a user-defined library of Simulink blocks and inputs for a NCS. In CDML, a set of modeling primitives and attributes such as T_s (Sampling Time), ModelName and ModelLibraryName etc., can be used to specify the model and parameters that define the control system components. From the defined parameters in CDML, an integrated interpreter in CDML generates Matlab code which when executed builds the desired Simulink models for implementing the control system with integrated HLA-based interfaces for run-time simulation in the HLA framework.

Example

In order to illustrate the CDML, we use the NCS example defined in Figure 48. Figure 51 shows a model of the CDML describing the control design model of the example. Unlike the model, in Figure 49, the Plant and Controller in this case represent the dynamics of the plant and the digital controller respectively. Using a set of modeling attributes defined in CDML, a user can specify the user-defined Matlab/Simulink model and parameters that define the dynamic behavior of a control system component.

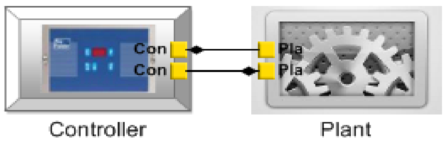


Figure 51: Control Design Model

6.3.2.3 Network Design Modeling Language

The Network Design Modeling Language (NDML) defines the modeling primitives for defining the dynamics of the communication network. This includes the capacity, loss models, routing and other additional properties to realize a desired networked control system. Figure 52 shows the meta model of the network design modeling language. From Figure 52, the main components are

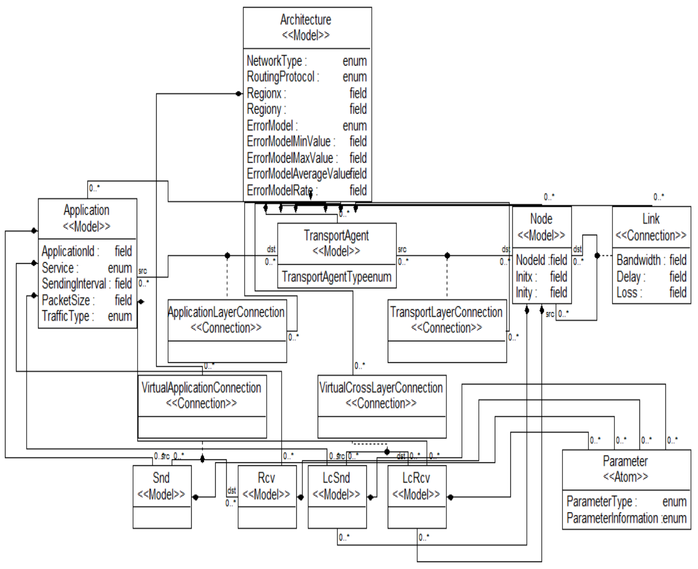


Figure 52: Network Design Meta Model

node, application, transport agent, link and the various connection types that exist between the components.

1. **Node** The node component represents the host or a computing unit on which an particular application (control design component) is running.

2. **Application** This models the actual application that runs at the top of the network stack. In our case this is an abstraction of the control design components.

3. **Transport Agent** This models the protocols or agents for delivering messages from the application layer of the source to that of the destination such as UDP or TCP.

4. **Link** This models the link layer of the network. It defines the path for delivering packets from a source node to a destination node.

It should be noted NDML does not re-implement the ns-2 syntax and simulation semantics. NDML essentially provides an interface language for defining the network interactions between the ns-2 simulation and the Matlab simulation. Examples of such interactions include the controller/plant deployment on the network node, traffic types etc. The ns-2 syntax and semantics that are internal to networking simulation (e.g., packet scheduling, routing) will not be reflected in NDML. We rather rely on the TCL language of ns-2 for the network internal simulation configuration.

Similar to CDML, a model transformation is used to transform the base architecture model in NCSWT MIL directly to a base model in NDML in order to preserve the consistency of the models in the different design views. Then the network primitives defined in NDML are used to define the network properties for the NCS. A user can then specify the transport agent, loss model of network links and other various network properties to simulate the network dynamics. Run-time network configuration and model scripts can then be generated using an integrated code generator in NDML based on the defined model parameters for deployment in ns-2 for the execution of the NCS simulation.

Example

In order to illustrate the NDML, we use the NCS example defined in Figure. 48. Figure 53 shows a model of the NDML describing the network design model of the example. This model specifies the transport agents, the network topology and other network properties required for the exchange of information between the control design components over a network. Unlike the model shown in Figure 49 and Figure 51, the Plant and Controller in this case represent the network applications. The TPAgent1 and TPAgent2 represent the transport agent models attached to the network hosts, Node1 and Node2 respectively. Using a set of modeling primitives and attributes in NDML, a user can specify the loss model of network links to simulate the desired network dynamics.

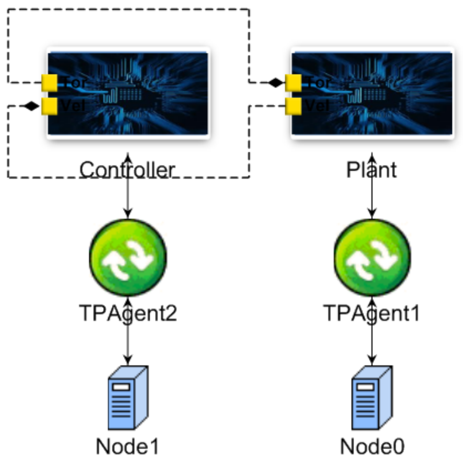


Figure 53: Network Design Model

6.3.3 Design Flow for the NCSWT Model-Based Approach

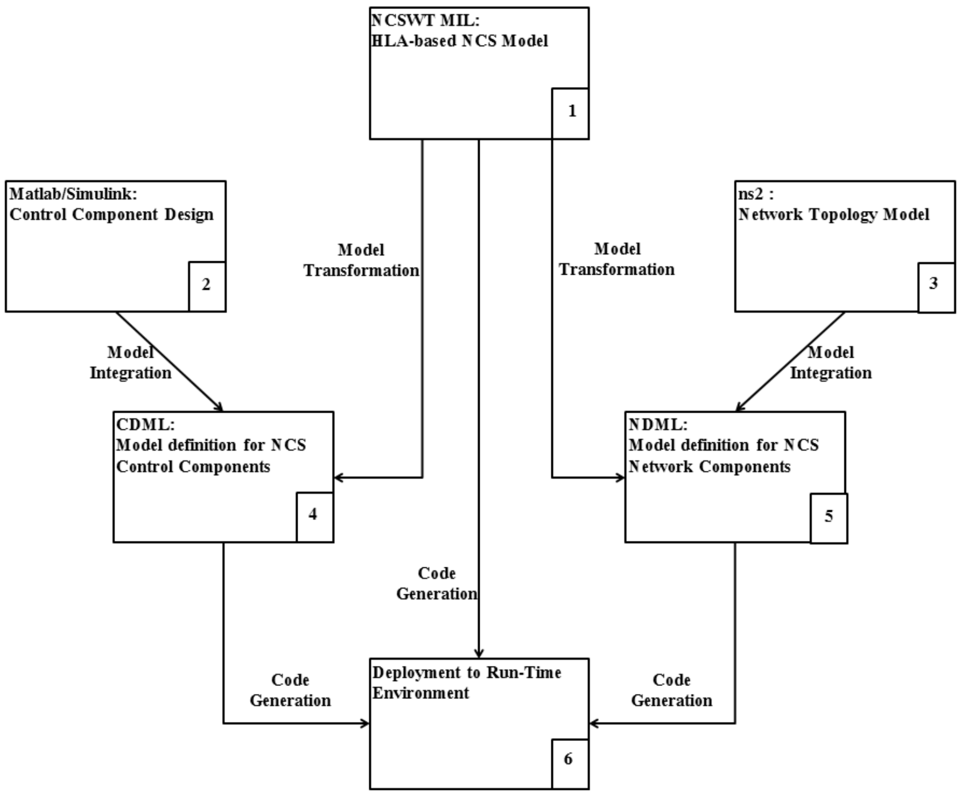


Figure 54: Model-Based Design Flow for NCSWT

Figure 54 shows the design flow for the NCSWT model-based approach. The number depicted on the bottom right of each of the blocks represents the design stage of the model-based approach.

- **Step 1:** In this step, a user defines the NCS to be simulated in the NCSWT MIL using HLA concepts that capture the control components and network components of the NCS as well as the information flow between the components. Using a set of integrated interpreters, two model transformations are executed on the NCSWT MIL to generate a CDML model and a NDML model for the definition of the control design and network design components. Additionally, configuration scripts and glue code required for the run-time simulation of the NCS are also generated from the NCSWT MIL model.

- **Step 2:** In this step, a user designs the control components of the NCS in Matlab/Simulink. Each individual component of the NCS that is to be simulated separately is stored in a user defined library. The reference name of the component model and the library name in which it is stored are used in the CDML. If the control component is a standard Simulink block that can be located in the Simulink library, the user determines the reference name of the block in the library as this is used in the CDML in order to build the NCS model.

- **Step 3:** Depending on the complexity of the network, a user can specify a network topology of the NCS in ns2 using a third-party tool such as GT-ITM for realistic Internet topology generation [329]. The reference name of this topology can be provided as an input to the NDML. Alternatively, a medium sized network topology can be modeled directly in NDML.

- **Step 4:** This step involves the modeling of control components using CDML. In CDML, starting from the transformed model from the NCSWT MIL, each of the control design components are captured through the use of attributes defined in the CDML. These attributes specify the user library containing the control component Simulink models as well as the reference block names for the components created in Step 2. An integrated interpreter traverses the CDML instance model and generates Matlab code which when executed generates Matlab/Simulink models for the control design components integrated with the HLA-based interfaces required for the NCS simulation.

- **Step 5:** In the NDML, starting from the transformed model from the NCSWT MIL, a user specifies the network components such as the transport agent, node and application agents for the NCS. These components together with their parameterized attributes capture the underlying network communication features for the NCS. A set of integrated interpreters traverses the model and generates tcl for the NCS model. Also, the HLA-interface scripts required for the integration of ns2 for the run-time simulation is also generated file.

- **Step 6:** The final step involves the deployment of the generated models, glue code and configuration files from the subsequent steps in the run-time environment to facilitate the actual simulation of the NCS using our framework.

6.4 NCSWT Run-Time Components

The Run-Time components of the NCSWT are shown in Figure. 55. These components represent the main software components and interfaces for the realization of a NCS simulation using the HLA framework. These components include the simulators (Matlab/Simulink and ns-2), the Run-Time infrastructure (RTI), the federates, and all the necessary glue code for the interfaces as well as monitoring tools for visualizing and evaluating the results.

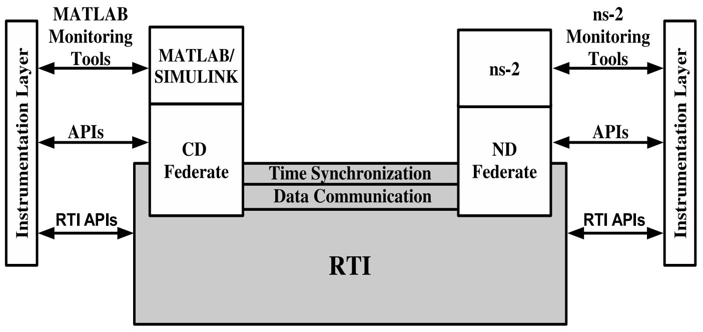


Figure 55: Run-Time Components

6.4.1 Run-Time Infrastructure (RTI)

The RTI, an implementation of the HLA standard, manages the communication between different federates. Using interactions, federates communicate between each other through the RTI [330]. The RTI handles the coordination of time and data passed between federates. A number of commercial and academic RTI implementations are available. Currently, we use Portico version 1.0.2, an open source cross-platform HLA implementation, which supports both C++ and Java clients [330].

Each federate has a single point of contact to the RTI through which it can communicate with other federates. Each federate represents a single instance of the corresponding simulator' interface to the RTI. For example, the ND federate is a software component that interfaces the ns-2 simulator with the RTI. The RTI provides a set of programmable application interfaces to instantiate a federate and for the federate to communicate with the RTI.

We briefly describe the NCSWT run-time services provided by the RTI.

6.4.1.1 Time Management

In an HLA-based federation, each federate has its own logical time. The RTI preserves the causality of the federation by ensuring that no simulation receives an event that occurred in the past relative to its own logical time. The RTI ensures the accurate progression of time through the use two main basic operations, the time advance request (TAR) and the time advance grant (TAG). In order for a federate to progress its logical clock during a simulation run, it must send a TAR to the RTI which then determines based on the current clocks of all the other federates in the federation whether to send a TAG for the federate to proceed. The RTI chooses the smallest of all the federates' TAR times, if a federate TAR time is larger than the granted TAG, it will block and it will only proceed its simulation when its TAG is successful [331].

6.4.1.2 Data Communication and Coordination

The RTI uses a publish-and-subscribe mechanism for passing messages through the federation in order to ensure the accurate data communication and coordination between the federates [331]. The type of messages exchanged between the federates are determined by the interactions and the interaction parameters defined in the NCSWT MIL. In the publish-and-subscribe mechanism, the sender of the interaction, known as the publisher, publishes the interactions to the RTI making it available to any receiver, known as the subscriber, registered to receive those kind of interactions. The RTI assures timely delivery of the interactions to all subscribing federates.

In addition to these services, in our framework the RTI provides additional services for monitoring the progression of a simulation during run-time.

6.4.2 Federates

In order to participate in a federation, the ND and CD federates utilize the services provided by the RTI. We briefly discuss how each of the federates utilize these services.

6.4.2.1 ND Federate

In order to participate in a federation, a ND federate, which interfaces the communication network simulated in ns-2 to RTI, leverages a set of generic classes defined by completely reusable C++ code. The reusable C++ code provides all of the fundamental RTI integration requirements such as converting between ns-2 defined types and RTI types, encapsulating and interfacing with the RTI for initializing the federate, synchronizing the simulator's clock and managing the publish-and-subscribe relationship with other federates.

The design-time models facilitate the integration of the ND federate in an HLA based federation. In our framework, a ND federate can be defined in the NCSWT MIL model, along with all its associated interaction types and parameters. This definition directly integrates the network design components simulated in ns-2 with an HLA-federation. From the NCSWT MIL, a GME interpreter is used to generated C++ files used during the run-time simulation of the NCS. The generated files leverage the reusable generic C++ classes together with the integrated interactions and parameters specification for the ND federate defined in the NCSWT MIL in order to participate in a federation.

Figure 56 shows a time synchronization example with the ND federate. Because the ns-2 simulator uses a discrete event model of computation, time synchronization can be incorporated along with event scheduling. In particular, the C++ source code for the ns-2 simulator scheduler is modified to implement the time synchronization mechanism defined by the RTI. The scheduler blocks the scheduling of new events until it receives an adequate time advance grant (TAG) from the RTI. The ns-2 scheduler submits a time advance request (TAR) once it is ready to execute an event scheduled for at a time later than the latest TAG. An initialization TAR is passed from ns-2 to the RTI, and once a TAG is received, the ns-2 scheduler can unblock and run until it reaches a synchronization event (an operation requiring synchronization with the RTI). At that point, a TAR is submitted to the RTI and ns-2 blocks. It is possible that the newest TAG will be less than the most recent TAR (i.e. $t_2 < t_1$), so ns-2 only executes events scheduled for times less than or equal to the latest TAG. Since our modification is made to the base class scheduler, our approach naturally supports the different types of Calendar Schedulers that are available in ns2.

For the ND federate's data communication, the communication network simulated in ns-2 uses a set of function calls defining the interactions associated with the federate, together with the reusable

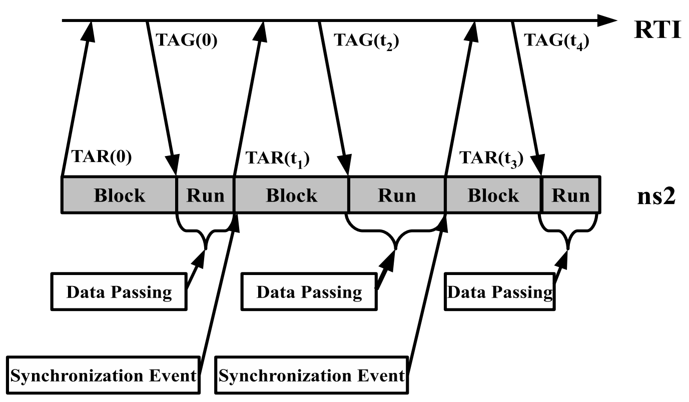


Figure 56: Time synchronization between the NS-2 federate and the RTI

generic C++ classes to send and receive interactions using the publish and subscribe mechanism provided by the RTI. The function calls are generated from a set of GME interpreters integrated in the NCSWT MIL and NDML. The interactions, which are modeled in the NCSWT MIL, represent the data exchanged between the ns-2 simulator and Matlab/Simulink.

6.4.2.2 CD Federate

In order to participate in a federation, a CD federate, which interfaces the control design components simulated in Matlab/Simulink to RTI, leverages a set of generic classes defined by completely reusable Java code. The reusable code provides all of the fundamental RTI integration requirements such as converting between Matlab/Simulink types and RTI types, encapsulating and interfacing with the RTI for initializing the federate, synchronizing the simulator's clock and managing the publish-and-subscribe relationship with other federates [186].

In our framework, a CD federate can be defined in the NCSWT MIL model, along with all its associated interaction types and parameters. This definition directly integrates the control design component simulated in Matlab/Simulink with a HLA-federation. From the NCSWT MIL, a GME interpreter is used to generated Java and Matlab/Simulink files used during the run-time simulation of the NCS. The generated files leverage the reusable generic classes [186], together with the integrated interactions and parameters specification for the federate defined in the NCSWT MIL, in order to participate in a federation.

The time synchronization mechanism for the CD federate is simulated in Matlab/Simulink using a Simulink S-function block integrated in Simulink model of each control design component.

Each control design component, using function calls in the S-function blocks, implements the time synchronization mechanism provided by the RTI [186]. These function calls implement the TARs/-TAGs mechanism for each of the Matlab/Simulink component. Using these function calls, the component can request the advancement of the simulator's logical clock and Using these function calls, the component can request the advancement of the simulator's logical clock and the simulator's execution is allowed to proceed only when the RTI grants the advancement of the simulator's logical clock.

For a CD federate, each corresponding control design component also uses the Simulink S-function block to either publish or subscribe interactions in order to send or receive data. The control design component, using function calls in the S-function blocks, implements the sending or receiving of interactions from the RTI.

The CDML modeling language, automatically generates the Simulink model for the corresponding control design component with the integrated blocks containing the S-function calls for implementing the data communication and time synchronization mechanisms.

6.5 Implementation Overview

The NCSWT tool is intended to provide users with a flexible, extensible and convenient tool for simulating NCS. The simulation of a NCS using the NCSWT tool requires two major steps. The first step involves the modeling of the NCS and the generation of all the necessary models, configuration scripts and glue code for the simulation of the NCS. The modeling of the NCS is performed in GME using the three DSMLs discussed in Section 6.3. The modeling and code generation is performed on a computer running a Windows operating system. The second step involves the deployment of the generated code and models and the execution of the simulation. The simulation is executed on a computer running a Linux operating system. Figure 57 shows pictorial representation of NCSWT Implementation for the described simulation steps.

6.6 Case Studies

We present two case studies to demonstrate our tool as well as show how it can be used to evaluate the impact of network effects such as time-varying delays and packet losses on the overall

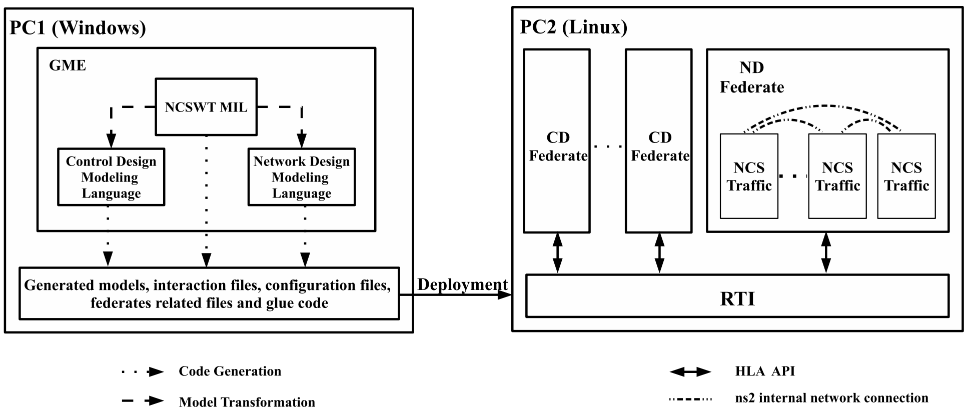

PC1 (Windows)
GME
NCSWT MIL
Control Design Modeling Language
Network Design Modeling Language
Generated models, interaction files, configuration files, federates related files and glue code
Deployment
PC2 (Linux)
CD Federate
CD Federate
ND Federate
NCS Traffic
NCS Traffic
NCS Traffic
RTI
Code Generation
Model Transformation
HLA API
ns2 internal network connection

Figure 57: NCSWT Implementation Overview

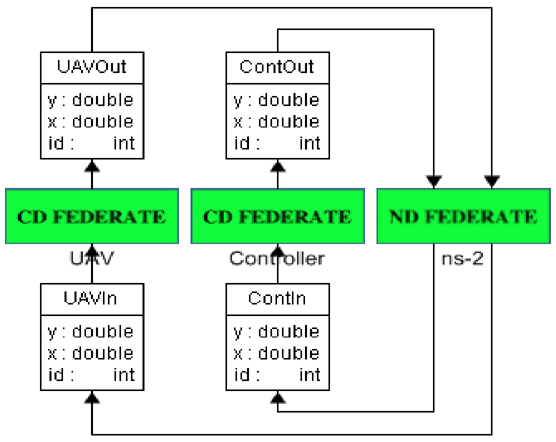
performance of a NCS.

6.6.1 Networked Unmanned Aerial Vehicle

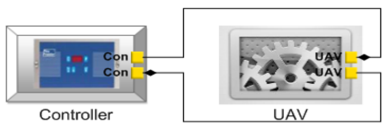
This NCS is comprised of an unmanned aerial vehicle (UAV) [14] controlled by a proportional derivative (PD) digital controller over a 802.11b wireless network. The networked digital controller, a proportional-derivative (PD) controller, is designed to enable the UAV to track a desired reference position trajectory. The main objective of this case study is to demonstrate the use of the NCSWT tool to simulate the impact of various network effects on the NCS. The UAV and the digital controller are modeled in Matlab/Simulink while the wireless network is modeled in ns-2. We consider three main scenarios: (1)Nominal case, (2) Scenario with a lossy network, (3) Scenario where multi-hop relay is employed for packet delivery. Figure. 58, shows the design-time models for the Networked Aerial Vehicle.

6.6.1.1 Nominal Case

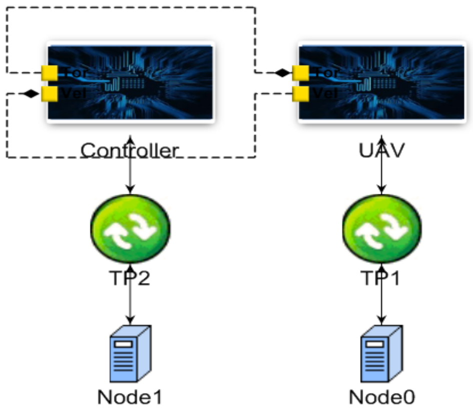
In this experiment, we simulate the NCS composed of the UAV and a digital controller communicating through a single hop wireless network. The UAV and the networked controller are located within the transmission range of each other and there are no additional network effects such as packet losses and delays. The sampling period of the networked controller is 0.1 seconds. Figure 59(a) shows a plot of the UAV x and y positions as well as the reference position trajectory. The plant tracks the reference trajectory so closely that the difference is imperceptible in the figure. Fig-



(a) NCSWT MIL Model



(b) CDML Model



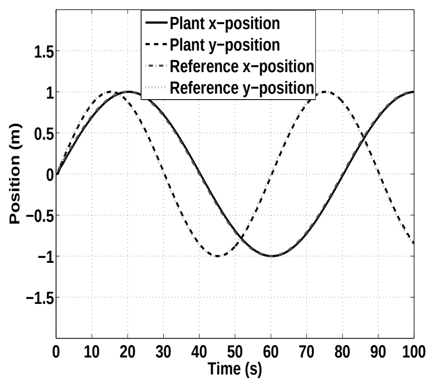
(c) NDML Model

Figure 58: Design-Time Models for Networked Aerial Vehicle.

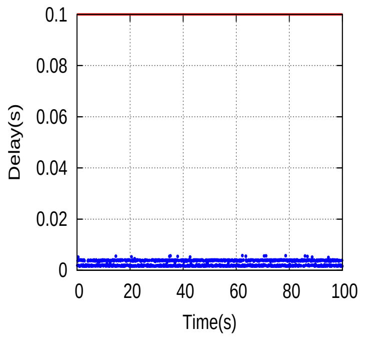
ure 59(b) shows the end-to-end delay plot for the Nominal case with the sampling period denoted by the horizontal line at 0.1s. It can be seen that the delay is much less than the sampling period, so it is to be expected that the plant is able to follow the reference signal closely.

6.6.1.2 Scenario with a Lossy Network

Wireless network communication can be unreliable, so in this experiment we demonstrate the impact of packet losses on the performance of the NCS when packets are lost in the communication channel of the network. We simulate a lossy communication channel based on a uniform probability distribution [332]. The results for the loss rates of 20% and 40% are presented in Figure 60. As the loss rate increases, the UAV trajectory strays further from the reference trajectories as can observed



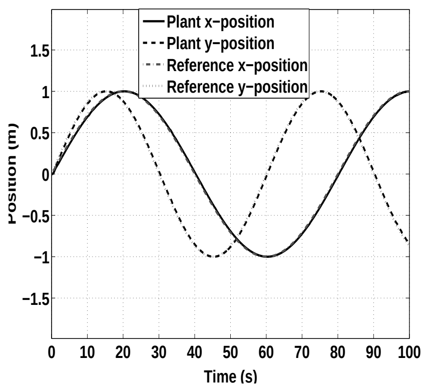
(a) UAV and Reference Trajectories.



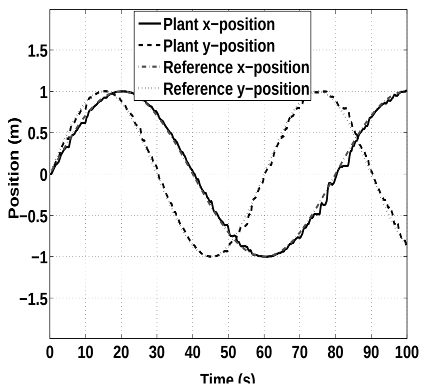
(b) End-to-End delay plots

Figure 59: Plots for the nominal case of the Networked UAV.

in Figure 60 which would be expected.



(a) Output plot for 20% loss rate.



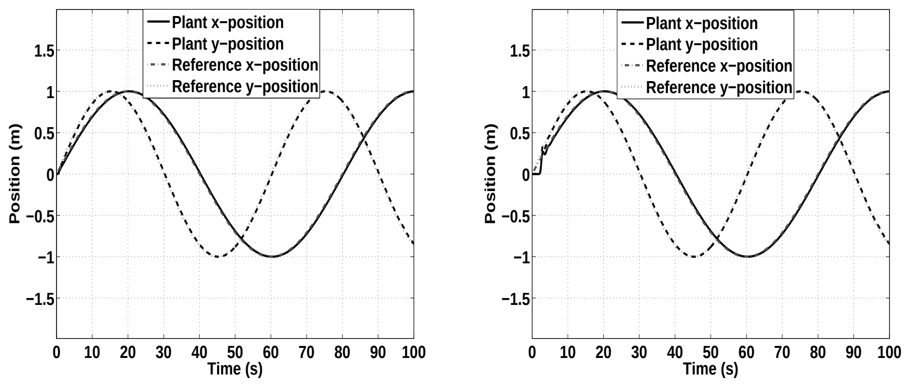
(b) Output plot for 40% loss rate.

Figure 60: Plots of UAV trajectory for packet loss rates.

6.6.1.3 Scenario where multi-hop relay is employed for packet delivery

The direct connection of two nodes in wireless networks requires the two nodes to be within the transmission range of each other however, this may not always be possible. In order to enable the communication of two nodes that are outside of each other's transmission range, intermediate nodes can act as relays to route the packets to their final destination nodes. Such network architecture are called wireless multi-hop networks. We use a chain topology with a static routing protocol to test the multi-hop scenario for the Networked UAV NCS. In this topology, nodes are formed in a chain structure with a fixed distance of 200m between neighboring nodes. The plant node and controller node are located at the edges of the network such that they will need to transmit information through intermediate nodes.

Figure 61, shows the simulation results for three and five chain networks. From Figure 61, it can be seen (especially at the beginning of the trajectory plots) that as the number of hops increase, the UAV trajectory deviates further away from the reference trajectory due to the increased delay introduced by the number of hops between the plant and controller nodes.



(a) Output plot for Three hops. (b) Output plot for Five hops.

Figure 61: UAV trajectory for multi-hop communication.

6.6.2 Industrial Networked Control System (INCS)

In this case study we demonstrate the ability of the NCSWT to handle asynchronous sampling times in large systems. Systems with asynchronous sampling rates can cause resource contention problems for the communication channel, so it is very important to accurately model the behavior. Additionally, since NCS can potentially be large in size it is also very important that the tool is able to handle NCS of varying sizes. By using the HLA communication standard, the NCSWT simulation tool appropriately handles the simulation of large NCS with various sampling rates as described in Section 6.4.

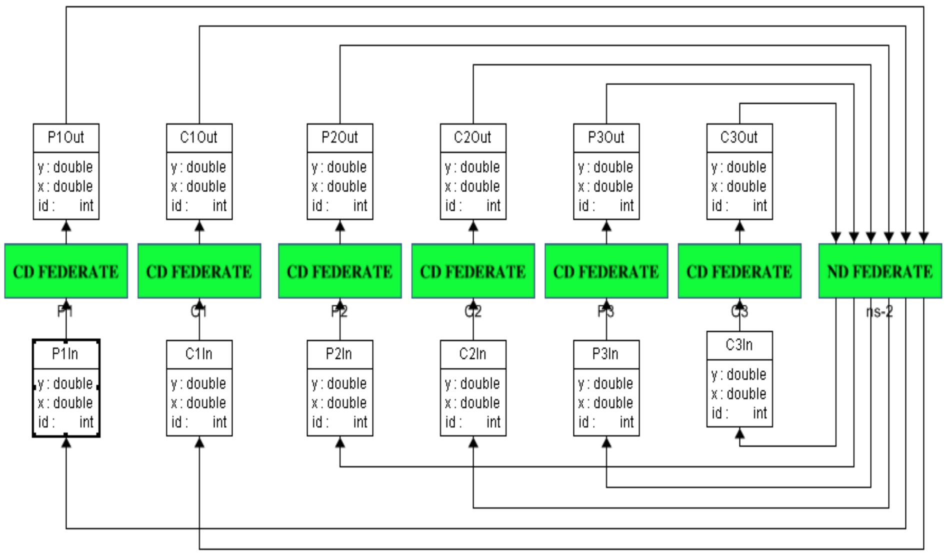
This case study is a typical industrial networked control system. It involves the simulation of three networked control systems working concurrently over the same wireless network. The networked control systems are denoted NCS1, NCS2 and NCS3, and they execute with the sampling times 0.1s, 0.15s and 0.25s respectively. Each of the networked control systems is composed of a plant system and a proportional derivative (PD) controller that controls the corresponding plant to behave in the desired manner. NCS1 is composed of the plant system P1 and controller C1; NCS2 is composed of the plant system P2 and controller C2 and NCS3 is composed of the plant system P3 and controller C3. The model for each of the plant systems, P1, P2 and P3, is an industrial robotic arm. The model for each of the controllers C1, C2 and C3 is identical except for the different sampling times. We consider two scenarios: (1) Nominal case, (2) Scenario with network effects. Figure 62, shows the design-time models for the INCS.

6.6.2.1 Nominal Case

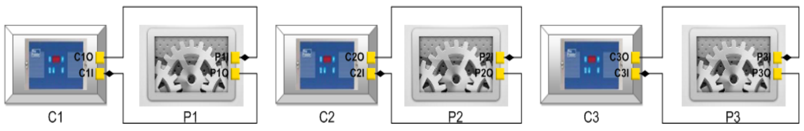
In this experiment, the three NCS are operating in a single-hop network without any additional network effects such as packet losses and delays. All six nodes use the same network channel. The reference trajectories for this experiment are essentially identical to the reference trajectory in the single UAV networked system in Section 6.6.1. Figure 63 shows the end-to-end delay plots. The horizontal red line in each plot indicates the sampling time for each plant-controller pair. It can be seen that the delays for each of the NCS are less than their corresponding sampling period.

6.6.2.2 Scenario with network effects

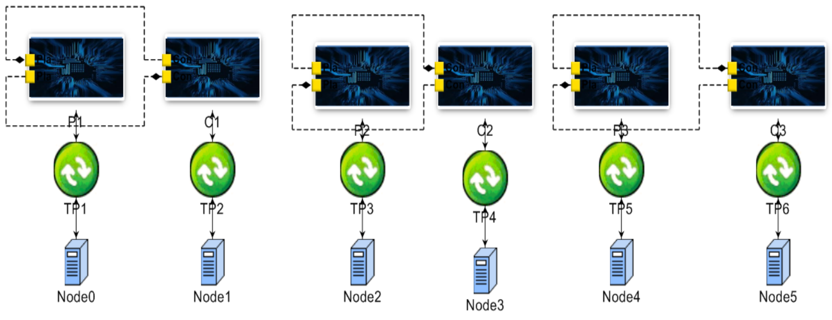
We simulate background traffic and packet loss in the wireless network to evaluate their impact of network effects on the overall performance of the INCS. Two nodes were introduced in the network to simulate background traffic while a uniform probability loss model [332] was used to simulate packet loss in the communication channel. Figure 64 shows a plot of the outputs of the plants showing the effect of the background traffic in addition to a 30% loss rate. It can be seen that the simulated network effects affect the outputs of the plants as depicted by degraded performance in tracking the reference trajectories. Figure 65 presents the network delay for packets transmitted between the plant and the controller for each of the three NCS. In the delay plots in Figure 65, the horizontal red line in each plot indicates the sampling time for each plant-controller pair. The figure shows similar delay for most of packets in the system, and this is reasonable since all the six nodes share the same network channel and therefore contend for network resources.



(a) NCSWT MIL Model



(b) CDML Model

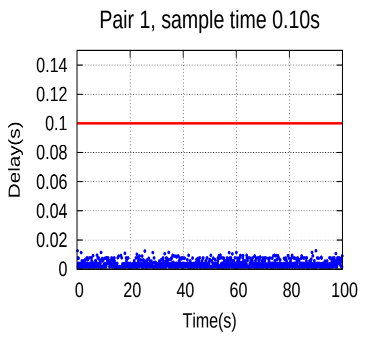


(c) NDML Model

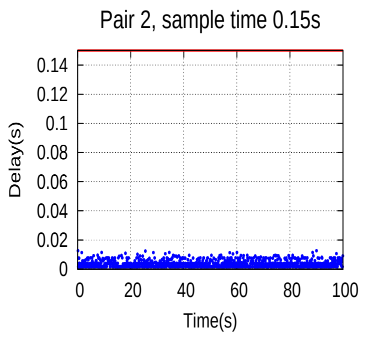
Figure 62: Design-Time Models for INCS.

6.7 Evaluation

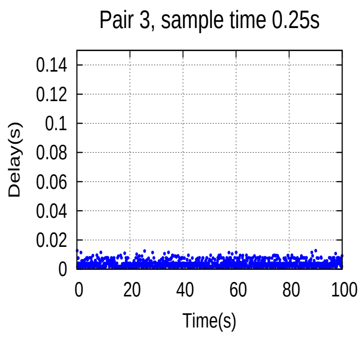
In order to build the NCSWT tool, the software packages shown in Table 5 are needed. Matlab/Simulink and ns-2 are used for the simulation of the control system and communication network of the NCS respectively. Portico 1.0.2 is the RTI implementation of the HLA used for running the federation. GME is the graphical environment used for the modeling and generation of all the necessary components for the simulation of the NCS. Universal Data Model (UDM) is utilized in



(a) Delay plot for P1-C1 pair



(b) Delay plot for P2-C2 pair



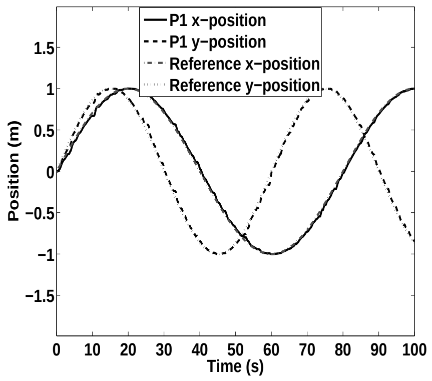
(c) Delay plot for P3-C3 pair

Figure 63: End-to-End delay plot for the nominal case scenario of INCS.

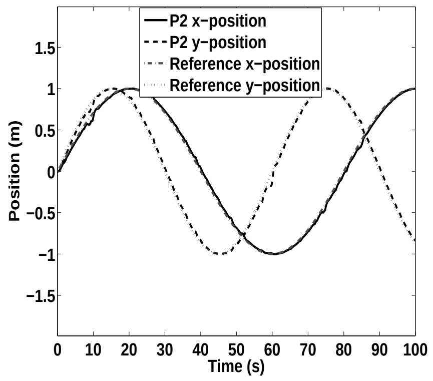
the model transformations from the NCSWT MIL to CDML and NDML. Microsoft Visual Studio is used for the execution of the code generators and model transformations from the three DSMLs. Eclipse is used for the compilation of the run-time components required for the simulation. The software packages ns-2, Portico 1.0.2, GME, UDM and Eclipse are freely available, while Matlab/Simulink and Microsoft Visual Studio are proprietary tools.

Table 5: Required Software Packages

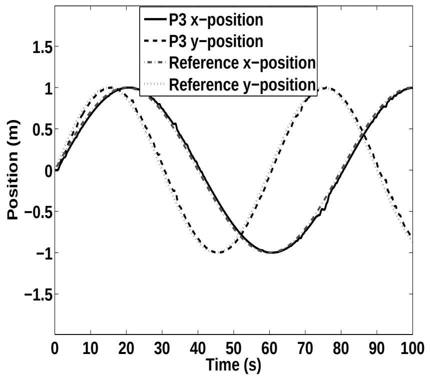
1. Matlab/Simulink, `www.mathworks.com`
2. ns-2, `http://isi.edu/nsnam/ns`
3. Portico 1.0.2, `www.porticoproject.org`
4. Generic Modeling Environment (GME), `www.isis.vanderbilt.edu/Projects/gme`
5. Universal Data Model (UDM), `http://www.isis.vanderbilt.edu/tools/UDM`
6. Microsoft Visual Studio 2008 or later, `www.microsoft.com/visualstudio`
7. Eclipse, `www.eclipse.org`



(a) Output plot for Plant P1.



(b) Output plot for Plant P2.

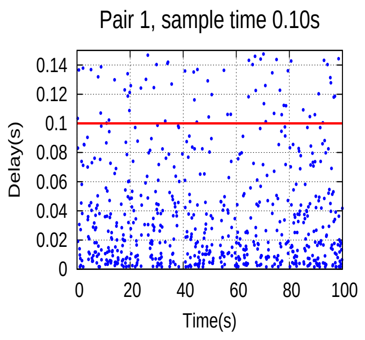


(c) Output plot for Plant P3.

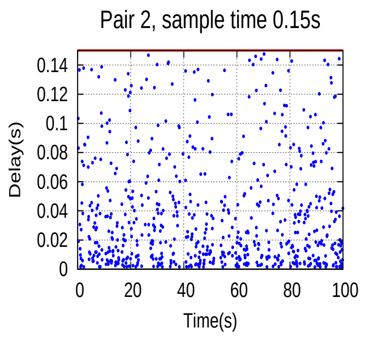
Figure 64: Output plots for the addition of background traffic and 30% packet loss in INCS.

NCSWT allows a user to rapidly reconfigure their experimental setup for the simulation of a NCS. For example, in order to evaluate the impact of network effects on a NCS, the configurations in the NDML is modified to the desired network configuration setup and then updated code is generated from the network model without modifying the control models. Similarly, if the dynamics of the control system is changed, the models in the CDML are modified to reflect the changes while maintaining the same network configurations, and then updated control models are generated.

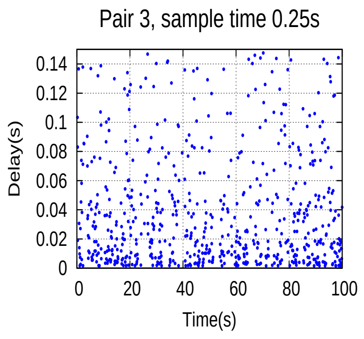
If the information passed between the control system and the network is changed, the NCSWT MIL model of the NCS needs to be modified to reflect the change. For example, when an additional parameter or new information type is introduced in the NCS, a new interaction type can be added to the existing NCS model defined in the NCSWT MIL and the new parameter can be added to it. If the new parameter needs to be added to an already existing information exchange type, the



(a) Delay plot for P1-C1 pair



(b) Delay plot for P2-C2 pair



(c) Delay plot for P3-C3 pair

Figure 65: End-to-End delay for the addition of background traffic and 30% packet loss in INCS.

parameter can be added to the existing interaction reflecting the information type that is exchanged between the federates. In a different case, if a new control component is introduced to the NCS, a new CD federate needs to be added to NCSWT MIL to reflect the introduced control design component and the required interactions are defined to reflect the communication of the new federate with the other federates. After the modifications to NCSWT MIL model of the NCS, update code and models are generated to reflect the changes to the model. In the case we need to modify the communication exchange between control components by adding new interactions or introduce additional components by adding federates, we use NCSWT MIL to add the necessary components and essentially repeat the NCS design process as described in Section 6.3.

We demonstrate the design-time efficiency for the Network UAV example. Recall, this NCS involves the digital control of an unmanned aerial vehicle (UAV), representing the plant, over the 802.11b wireless network to track a desired trajectory. For the design-time efficiency, we consider the amount of code that is automatically generated for simulating the NCS. Table 6 provides a summary of the size of code and models that are automatically generated from the design-time models. For the run-time efficiency, we consider the actual time it takes to simulate the NCS.

Table 6: Generated Code for Networked UAV Case Study

Files	**Size**
1. Matlab models	100 Kilobytes
2. Matlab glue code	132 Kilobytes
3. ns-2 model and topology scripts	20 Kilobytes
4. ns-2 glue code	160 Kilobytes
5. Federation startup script	4 Kilobytes

Table 7 shows the time durations for simulating the NCS in various multi-hop network topologies and in the presence of network uncertainties such as packet loss and time varying delays. The time durations shown in Table 7 are the actual times required to run 100 seconds of logical simulation time.

Table 7: Time Efficiency for Networked UAV Case Study

Scenarios		**Actual Duration (in minutes)**
Nominal		5.5
Packet Losses	20%	8.4
	30%	11.4
	40%	13.5
Multi-hop Network	3 hops	17.4
	4 hops	22.2
	5 hops	26.2

We also provide the run-time efficiency of the multi-agent NCS (MANCS) example. Similar to the networked UAV case, the time durations shown in Table 8 are the actual times required to run 100 seconds of logical simulation time.

Table 8: Time Efficiency for MANCS Case Study

Scenarios		**Actual Duration (in minutes)**
Nominal		10.0
Background Traffic and 30% Packet Loss	3 hops	17.4
	4 hops	22.2
	5 hops	26.2

6.8 Summary

The design and analysis of NCS is a critical task due to the complex interactions and uncertainties introduced by network effects. Simulation is a powerful technique in evaluating various NCS models and the impact of network effects on the overall system performance. In this Chapter, we present an integrated modeling and simulation tool, NCSWT, for NCS. We describe the HLA-based approach guiding the tool's implementation as well as the MIC techniques for the rapid synthesis of components required for the simulation of a NCS. We demonstrate the capabilities of the tool using case studies and also provided an evaluation of tool.

Our use of HLA-based framework provides great opportunities in regards to extending the framework to other simulators. The approach is modular and extensible. Based on previous works in [186, 327, 328] as well as the NCSWT, which involve the integration of various simulators using HLA, the inclusion of other simulators such as ns3, opnet, Modelica etc. can follow a similar approach. The main challenges involve identifying the critical sections in the simulators to integrate the time synchronization mechanism and data distribution services offered by the HLA standard.

CHAPTER 7

ENERGY-BASED ATTACK DETECTION IN NETWORKED CONTROL SYSTEMS

7.1 Introduction

The increased autonomy of CPS, together with the introduction of communication networks, has increased the security vulnerabilities of CPS infrastructure to intentional and malicious cyber attacks. Within the past few years, there has been a surge in attacks on CPS infrastructures. This increased prevalence of attacks has resulted in increased concerns regarding the security of these systems. Due to the safety-critical nature of CPS, failure or disruption of normal operation can potentially lead to serious harm to the physical system under control and to the people and other infrastructures that depend on it. Hence, securing these systems in order to ensure resilient operation is of utmost importance. Some of the well-known examples of attacks on CPS include the W32.Stuxnet worm attack that maliciously infected an Iranian Nuclear facility, taking control and heavily disrupting its normal operation according to the attacker's design [333], the cyber attacks on power transmission networks operated by Supervisory Control and Data Acquisition (SCADA) Systems [334], as well as attacks that infiltrated critical systems including medical devices [335], waste water treatment plants [336] and NRG generation plants [337].

In securing and ensuring the safety of CPS infrastructures, the reliable detection of attacks is very important. It is also fundamental to the design of compensation and reconfiguration mechanisms for mitigating the impact of attacks. Attack detection has become a very active research with numerous techniques being sought to facilitate the early detection of attacks on CPS. The presence of the network increases the complexity of the detection of attacks. Hence, effective and yet efficient novel approaches are needed to enable the early detection of attacks in CPS. A majority of the existing detection approaches are typically from the cyber-security community. As highlighted in [226], the traditional approach often used in information/cyber-security neglects the knowledge of the physical process under control in the detection of attacks. Contrary to the traditional cyber-security approach, newer approaches in the CPS community, instead of creating models of network traffic or software behavior, leverage the knowledge of the physical process in designing effective mechanisms in order to facilitate the detection of attacks. The idea is that by understanding the interactions of the control system with the physical world, it would be possible to develop systematic frameworks to detect attacks and secure CPS in general.

In this work, we utilize the properties of physical systems in order to define precise detectability conditions for certain attack models and vulnerabilities. An example of such physical system property is energy. The concept of energy is very important in the behavior of dynamical systems and it is typically common to view dynamical systems as energy-transformation devices. The concept of energy can also be extended in certain cases where an abstract notion of energy can be constructed. As discussed in Chapter 2, there has only been a handful of works in the area that use the concept of energy in addressing the problem of detection and most of these works have been focused on reliability as it pertains to the protection of physical components against faults. Additionally, existing work does not consider the introduction of a communication network and do not address the detection of intentional malicious cyber attacks against CPS.

In this chapter, using the intuitive notion of energy, we propose an attack detection mechanism for CPS. The proposed approach is complementary to other detection mechanisms such as those described in Chapter 2 (e.g. observer-based detection). The underlying idea is that the presence of attacks disturbs the energy balance of the physical system by dissipating or injecting additional energy. We define the notion of detectability of an attack by its effect on system's energy. Compared to the existing detection techniques, the proposed approach provides the additional benefit of detecting when and to what magnitude a system's energy property is impacted due to the occurrence of an attack. Based on this characterization, one can estimate the impact of the attack on the stability of the overall system. In particular, we focus on dissipative CPS, which include a large class of existing systems. We present the use of energy-balance in the detection of attacks in NCS. We present a general characterization of attacks on the energy of a dynamical system and also we demonstrate the impact of specific attack models on the stability guarantees of NCS. Finally, we demonstrate our approach using a case study on the tracking control of a single joint of a robotic arm over a network.

The contributions of this work are summarized as follows:

1. We formulate the detection of attacks using the system property of energy. We define the

notion of energy-based monitor for networked control systems and introduce the Energy-Based Attack Detector (EBAD) whose mechanism is based on a system's energy.

2. We prove that EBAD solves the attack detection problem. In addition, we illustrate that the impact of attacks on a system can be estimated as the excess or loss in a system's energy. Based on the estimated energy, attacks can be qualitatively characterized as either being passive or non-passive.

3. Based on well-known attack models, we illustrate the impact of the defined attacks on the energy of a system using the formulation of EBAD. We also present analytical results to show conditions in which the attacks can violate passivity properties of the overall system.

4. We evaluate the proposed methodology using simulations as well as experiments on a robotic manipulator test-bed.

The rest of the chapter is organized as follows. The networked control system model, the attack models and problem statement are given in Section 7.2. The energy-based attack detection approach, the analytical results on the detection mechanism and the characterization of passive and non-passive attacks are presented in Section 7.3. Section 7.4 presents an evaluation of the proposed approach using simulations. An experimental case study on an actual robotic system testbed is presented in Section 7.5 in order to illustrate the application of the approach to a physical system. Finally, a summary of the chapter is presented in Section 7.6.

7.2 System Model and Problem Statement

In this section, we describe the components of networked control system and the attack models considered in this work. Subsequently, we formulate the attack detection problem and describe the underlying assumptions. The notations used in the following sections are standard. Let $\mathbb{R}^n$ denote the Euclidean space of dimension n, I denotes the identity matrix of appropriate dimensions. For a matrix $P \in \mathbb{R}^{n \times n}$, its transpose is denoted by P^T. For a symmetric matrix, P, where $P = P^T, \quad P > 0$ denotes it is positive definite. The norm of a vector or a matrix is given by $\| P \|$.

7.2.1 Networked Control System Model

We consider a networked control system as depicted in Figure 66. The main components of the NCS are the physical plant, the controller, the wave transformation (a static local controller), and the communication network. The data exchange between the plant and the controller is done over a communication network.

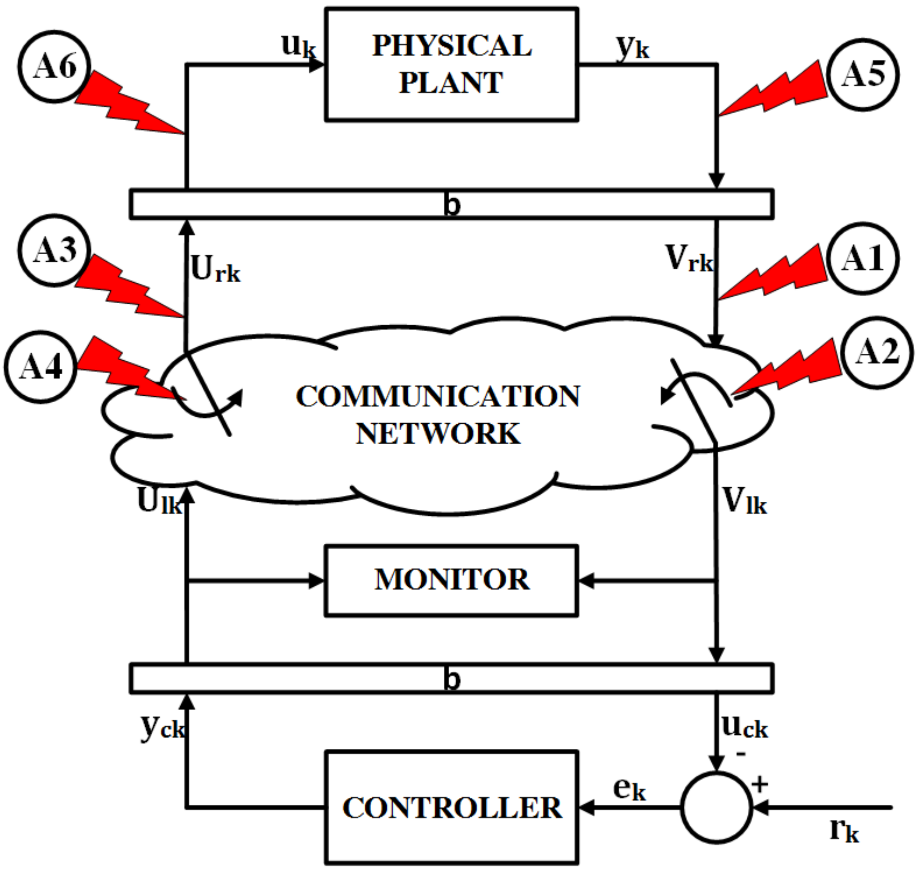


Figure 66: Networked Control System

(a) **Physical Plant Model**: We model the physical plant as a discrete time-invariant system. This version of the plant neglects system nonlinearities and presence of noise in the dynamics and measurement signals. We consider the physical plant which can be represented in the state space form as follows:

$$\mathcal{H}_p : \begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k + Du_k \end{cases} \tag{105}$$

where $x_k \in \mathcal{X}$ represents the state variables, $u_k \in \mathcal{U}$ represents the control inputs to the plant and $y_k \in \mathcal{Y}$ represents the plant outputs obtained by sensors at sampling instant $k \in \mathbb{Z}$.

(b) **Controller Model**: The controller modifies the behavior of the physical plant through the application of a control input command in order to achieve a desired objective or satisfy a performance requirement. The controller can be represented in discrete-time state-space form as follows:

$$\mathcal{H}_c : \begin{cases} z_{k+1} = A_c z_k + B_c e_k \\ y_{c_k} = C_c z_k + D_c e_k \end{cases} \tag{106}$$

where $z_k \in \mathbb{R}^q$ represents the controller states, y_{c_k} is the control command, $e_k = r_k - u_{c_k}$ is the error between the reference, r_k and the received plant output, u_{c_k}, with the matrices A_c, B_c, C_c and D_c of appropriate dimensions. It is assumed that the controller is designed under the nominal conditions, i.e. without attacks, to achieve the desired performance objective.

(c) **Wave Transformation**: In Section 3.6, we introduced the use of wave variables for exchange of information over the network to preserve the power content of the exchanged variables. From Figure 66, the wave transformation is denoted by the blocks, **b**. The wave variables pair (U_{r_k}, V_{r_k}) on the plant side as well as the pair (U_{l_k}, V_{l_k}) on the controller side of the network can be described by the following expressions:

$$U_{r_k} = \frac{1}{\sqrt{2b}}(y_k + bu_k) \tag{107}$$

$$V_{r_k} = \frac{1}{\sqrt{2b}}(y_k - bu_k) \tag{108}$$

$$U_{l_k} = \frac{1}{\sqrt{2b}}(u_{c_k} + by_{c_k}) \tag{109}$$

$$V_{l_k} = \frac{1}{\sqrt{2b}}(u_{c_k} - by_{c_k}) \tag{110}$$

where $b \in \mathbb{R}_0^+$. From Figure 66, one can observe that under ideal network conditions, $V_{l_k} = V_{r_k}$ and $U_{r_k} = U_{l_k}$.

Table 9: Attack Models and Compromised Components of NCS

Attack Type		**Wave Variables** (V_{r_k}, U_{r_k})	**Sensor** (y_k)	**Actuator**(u_k)
Integrity	*Min/Max*	**C**	**C**	**C**
	Additive	**C**	**C**	**C**
	Min/Max Energy	**C**	**NC**	**NC**
DoS		**C**	**NC**	**NC**

7.2.2 Attack Model

Figure 66 depicts the feasible cyber-attacks as a result of the vulnerabilities of the networked control system. While the attacks denoted as **A1**-**A4** model attacks on the information exchanged over the communication network, the attacks denoted as **A5** and **A6** models attacks on sensors and actuators respectively. Similar well-known attack types have be proposed in [228][227]. For each attack type, $\mathcal{A}_k$, let $\mathcal{T}_a = k_s, ..., k_e$ denote the attack duration with the attack starting from k_s and ending at k_e. We consider two main classes of attacks, integrity attacks and denial-of-service attacks. Table 9 provides a summary of the attack models considered in this framework. In the table, **C**, indicates the attack models considered for the corresponding components while **NC** indicates that the attack type is not considered. These attack types are described as follows:

(a) **Integrity attacks**: In an integrity attack, an adversary deceives a compromised component of the NCS into believing that a received false data is valid or true. The underlying assumption is that all attacks lie within a predetermined range since attacks leading to signals that exceed such a range can be easily detected. The integrity attacks represented as **A1**, **A3**, **A5** and **A6** in Figure 66 can be further categorized into the following:

  (i) *Min and Max attacks*: These attacks involve the adversary modifying the content of compromised signals to their respective minimum or maximum values. We model min/max attacks on the exchanged wave variables as well as the min/max attacks on the sensors and actuators. The attacks on the exchanged variables essentially exploit the vulnerabilities as a result of the communication while the attacks on the sensors and actuators exploit the vulnerabilities of the computing interfaces to these components which may or may not be colocated. We consider them separately since each component's interaction with the overall NCS is different and hence it is important to understand the impact of an attack on each component on the correct operation of the overall NCS.

(1) *Min/Max attacks on exchanged wave variables*

For attacks on the wave variable, V_{r_k}, sent from the plant we have,

$$\tilde{V}_{r_k}^{min} = \begin{cases} V_{r_k} & \forall k \notin \mathcal{T}_a \\ V_{r_{min}} & \forall k \in \mathcal{T}_a \end{cases} \tag{111}$$

$$\tilde{V}_{r_k}^{max} = \begin{cases} V_{r_k} & \forall k \notin \mathcal{T}_a \\ V_{r_{max}} & \forall k \in \mathcal{T}_a \end{cases} \tag{112}$$

Similar attacks can be launched against the wave variable, U_{r_k}, sent from the controller

$$\tilde{U}_{r_k}^{min} = \begin{cases} U_{r_k} & \forall k \notin \mathcal{T}_a \\ U_{r_{min}} & \forall k \in \mathcal{T}_a \end{cases} \tag{113}$$

$$\tilde{U}_{r_k}^{max} = \begin{cases} U_{r_k} & \forall k \notin \mathcal{T}_a \\ U_{r_{max}} & \forall k \in \mathcal{T}_a \end{cases} \tag{114}$$

(2) *Min/Max attacks on sensors and actuators*

For attacks on the sensor signal, y_k, we have,

$$\tilde{y}_k^{min} = \begin{cases} y_k & \forall k \notin \mathcal{T}_a \\ y_{min} & \forall k \in \mathcal{T}_a \end{cases} \tag{115}$$

$$\tilde{y}_k^{max} = \begin{cases} y_k & \forall k \notin \mathcal{T}_a \\ y_{max} & \forall k \in \mathcal{T}_a \end{cases} \tag{116}$$

Similar attacks could be launched against the actuator signal, u_k

$$\tilde{u}_k^{min} = \begin{cases} u_k & \forall k \notin \mathcal{T}_a \\ u_{min} & \forall k \in \mathcal{T}_a \end{cases} \tag{117}$$

$$\tilde{u}_k^{max} = \begin{cases} u_k & \forall k \notin \mathcal{T}_a \\ u_{max} & \forall k \in \mathcal{T}_a \end{cases} \tag{118}$$

(ii) *Additive attacks*: This attack involves introducing an additional offset/bias, $\alpha \neq 0$ to the actual exchanged information. We model additive attacks on the exchanged wave variables as well as additive attacks on the sensors and actuators.

(1) *Additive attacks on exchanged wave variables*

For attacks on the wave variable, V_{r_k}, sent from the plant we have,

$$\tilde{V}_{r_k}^a = \begin{cases} V_{r_k} & \forall k \notin \mathcal{T}_a \\ V_{r_k} + \alpha_k & \forall k \in \mathcal{T}_a \quad and \quad V_{r_k} + \alpha_k \in \mathcal{V} \\ V_{r_{min}} & \forall k \in \mathcal{T}_a \quad and \quad V_{r_k} + \alpha_k < V_{r_{min}} \\ V_{r_{max}} & \forall k \in \mathcal{T}_a \quad and \quad V_{r_k} + \alpha_k > V_{r_{max}} \end{cases} \tag{119}$$

Similar attacks can be launched against the wave variable, U_{r_k}, sent from the controller

$$\tilde{U}_{r_k}^a = \begin{cases} U_{r_k} & \forall k \notin \mathcal{T}_a \\ U_{r_k} + \alpha_k & \forall k \in \mathcal{T}_a \quad and \quad U_{r_k} + \alpha_k \in \mathcal{U} \\ U_{r_{min}} & \forall k \in \mathcal{T}_a \quad and \quad U_{r_k} + \alpha_k < U_{r_{min}} \\ U_{r_{max}} & \forall k \in \mathcal{T}_a \quad and \quad U_{r_k} + \alpha_k > U_{r_{max}} \end{cases} \tag{120}$$

(2) *Additive attacks on sensors and actuators*

For attacks on the sensor signal, y_k, we have,

$$\tilde{y}_k^a = \begin{cases} y_k & \forall k \notin \mathcal{T}_a \\ y_k + \alpha_k & \forall k \in \mathcal{T}_a \quad and \quad y_k + \alpha_k \in \mathcal{Y} \\ y_{min} & \forall k \in \mathcal{T}_a \quad and \quad y_k + \alpha_k < y_{min} \\ y_{max} & \forall k \in \mathcal{T}_a \quad and \quad y_k + \alpha_k > y_{max} \end{cases} \tag{121}$$

Similar attacks could be launched against the actuator signal, u_k

$$\tilde{u}_k^a = \begin{cases} u_k & \forall k \notin \mathcal{T}_a \\ u_k + \alpha_k & \forall k \in \mathcal{T}_a \quad and \quad u_k + \alpha_k \in \mathcal{U} \\ u_{min} & \forall k \in \mathcal{T}_a \quad and \quad u_k + \alpha_k < u_{min} \\ u_{max} & \forall k \in \mathcal{T}_a \quad and \quad u_k + \alpha_k > u_{max} \end{cases} \tag{122}$$

(iii) *Min/Max energy attacks*: Considering that the proposed approach is based on energy, an attacker's objective could be to apply the largest impact damage on the system based on the knowledge of the system's energy. We model two types of energy-based attacks based on their intended impact on the system.

(1) *Max energy attack*: In this case, we model attacks that attempt to dissipate maximum amount of energy i.e. the energy of the system becomes positive. This type of attack can be seen as an attacker's attempt to degrade system performance without destabilizing the system in regards to energy. In this attack type, for each time step, the attacker chooses a value for the compromised wave variable such that the total dissipated energy is maximized without exceeding the predetermined limits of the wave variable. The max energy attack can be captured as follows:

$$\begin{aligned} &\underset{V_{r_k}}{\text{maximize}} && E_T \\ &\text{subject to} && V_{r_k} \in [V_{r_{min}}, V_{r_{max}}] \end{aligned}$$

(2) *Min Energy Attack*: Similar, to the max energy attack, in this case we model attacks that attempts to inject the largest amount of energy which from the system's perspective portrays the system as generating additional energy i.e. the energy of the system becomes negative. This attack type can be seen as an attacker's attempt to both degrade the performance of the system and potentially destabilize the system. In the model of this attack, at each time step the attacker chooses the compromised wave variable such that the energy is minimized without exceeding the predetermined lim-

its of the wave variable. The min energy attack can be captured as follows:

$$\begin{aligned} \underset{V_{r_k}}{\text{minimize}} \quad & E_T \\ \text{subject to} \quad & V_{r_k} \in [V_{r_{min}}, V_{r_{max}}] \end{aligned}$$

(b) **Denial-of-Service (DoS) attacks**: DoS attacks, denoted as **A2** and **A4** in Figure 66, prevent signals from reaching the intended destination. In NCS, it involves the disruption of the availability of information exchanged between the plant and the controller. DoS attacks are typically carried out by jamming the communication channel, changing the routing protocol or saturating the receiver with useless signals. The attacker's main objective is usually to degrade the performance of the NCS as well as to potentially destabilize the physical system. The DoS attack can be modeled as a form of the additive attack as follows:

$$\tilde{V}_{r_k}^{DoS} = V_{r_k} + \alpha V_{r_k} \quad \begin{cases} \alpha = 0 \quad \forall k \notin \mathcal{T}_a \\ \alpha = -1 \quad \forall k \in \mathcal{T}_a \end{cases} \tag{123}$$

Similarly, for the wave variable sent from the controller, we have

$$\tilde{U}_{r_k}^{DoS} = U_{r_k} + \alpha U_{r_k} \quad \begin{cases} \alpha = 0 \quad \forall k \notin \mathcal{T}_a \\ \alpha = -1 \quad \forall k \in \mathcal{T}_a \end{cases} \tag{124}$$

7.2.3 Problem Statement

Consider the networked control system as shown in Figure 66, under possible cyber attacks as indicated by the attacks **A1**-**A6** due to the vulnerabilities of NCS. We define what is meant by an energy-based monitor and detectability of attacks in this framework.

Definition 7.1. *An energy-based monitor is a deterministic algorithm, $\Phi : \Lambda \mapsto \Psi$, with knowledge of the plant dynamics and access to discrete-time measurements and control inputs. The output of a monitor is $\Psi = \{\psi_1, \psi_2\}$, with $\psi_1 \in \{True, False\}$, and $\psi_2 \in \{Passive, Non - Passive\}$*

Definition 7.2. *An attack is detectable if in the presence of the attack, $\mathcal{A}_k$, ψ_1 =True and ψ_2 = Passive or Non-Passive.*

The following problem is of interest:

1. {**Detection Problem**} *Design an algorithm, Φ, for an energy-based monitor which can quantify or estimate the energy of the system, E_T, such that in the presence of an attack and with the knowledge of the plant, the controller and exchanged wave variables the following holds:*

$$\Psi = \{\psi_1, \psi_2\} = \begin{cases} \{\text{True}, \text{Passive}\} & \forall E_T > 0 \\ \{\text{True}, \text{Non-Passive}\} & \forall E_T < 0\} \end{cases} \tag{125}$$

In the following sections, we propose a solution to the above problem. We assume the following about the NCS.

Assumption 1: The plant and controller are dissipative by design, both with a sampling period, T$_s$. The assumption of dissipativity for both the plant and controller is to ensure stability guarantees in the nominal case.

Assumption 2: The components of the NCS including the physical plant, the sensor, actuator, controller and attack monitor are time-synchronized. This ensures that all the components of the NCS are progressing in lock step in regards to time.

Assumption 3: Whenever the input buffers are empty, null packets are processed. This assumption is used to preserve passivity in the nominal sense in order to avert the typical hold-last sample approach which is known to be non-passive. Other approaches for handling missed packets can be sought in this case as well with no loss of generality.

Assumption 4: It is assumed that the controller and monitoring system for the plant are co-located together. The idea is that the controller is assumed to be trustworthy while the plant's trustworthiness is not known or guaranteed. In the case that the trustworthiness of the controller is not known or guaranteed, an additional monitor can be co-located with the plant.

Assumption 5: The attacker has knowledge of the plant and controller. In this assumption, we consider that the attacker is smart in the sense that he/she can attempt to use knowledge of the system to introduce attacks that cannot be easily detected with a simple bad data detector.

Assumption 6: For our initial analysis, we assume an ideal communication network, hence do not consider the usual communication network effects such as time-delays and packet losses but rather we focus on malicious attacks on the cyber-physical infrastructure. In this regard, we assume that any anomaly in the behavior of the overall system is due to an attack. This assumption will be relaxed later to include network effects in our approach.

7.3 Energy-Based Attack Detection

In this section, we derive the energy balance for the networked control system in terms of the input-output wave variables, U_{r_k} and V_{r_k}. Next, we provide a generalized characterization of attacks based on the derived energy balance. We then evaluate the impact of the attack models presented in Section 7.2.2. Finally, we consider the case where the states of the system are not measurable, in which case we introduce the use of an observer to estimate the states.

7.3.1 Discrete-Time Energy Balance Derivation for NCS

We present the energy-based attack detection mechanism for the networked control system in Figure 66. We first present the derivation of general energy balance in terms of the plant's input, u_k and output, y_k, and then we refine the derivation to represent the energy balance system in terms of the wave variables exchanged over the network.

Proposition 7.3. *Consider the discrete-time physical plant, $\mathcal{H}_p$, with a minimal realization (controllable and observable) defined in (105). If $\mathcal{H}_p$ is QSR dissipative then it satisfies the energy balance, E_T given by*

$$E_T = E_{su} - E_{st} - E_d = 0 \tag{126}$$

where E_{su} is the supplied energy, E_{st} is the stored energy and E_d is the dissipated energy.

Proof. Recall the storage function, V_k, defined as $\frac{1}{2}x_k^T P x_k$. The change in the storage function, ΔV is given by

$$\begin{aligned}\Delta V &= V_{k+1} - V_k \\ &= \frac{1}{2}x_{k+1}^T P x_{k+1} - \frac{1}{2}x_k^T P x_k\end{aligned}$$

substituting x_{k+1} from (105), we have

$$\begin{aligned}\Delta V &= \frac{1}{2}((x_k^T A^T + u_k^T B^T)P(Ax_k + Bu_k) - x_k^T P x_k) \\ &= \frac{1}{2}(x_k^T (A^T PA - P)x_k + x_k^T A^T PBu_k + u_k^T B^T PAx_k + u_k^T B^T PBu_k) \end{aligned} \tag{127}$$

From the Generalized KYP lemma described in lemma 3.20, we can substitute (36), (37) and (38) into equation (127), then we have

$$\begin{aligned}\Delta V = \frac{1}{2}(x_k^T (C^T QC - LL^T)x_k + x_k^T (C^T QD + C^T S - LW)u_k \\ + u_k^T (D^T QC + S^T C - W^T L^T)x_k + u_k^T (R + D^T S + S^T D + D^T QD - W^T W)u_k)\end{aligned} \tag{128}$$

After some manipulation and simplification, we have

$$\begin{aligned}\Delta V =& \frac{1}{2}((Cx_k + Du_k)^T Q(Cx_k + Du_k) + 2(Cx_k + Du_k)^T Su_k + u_k^T Ru_k) \\ &- \frac{1}{2}(x_k^T LL^T x_k + x_k^T LWu_k + u_k^T W^T L^T x_k + u_k^T W^T Wu_k)\end{aligned}$$

From (105), noting that $y_k = Cx_k + Du_k$, we can now write

$$\begin{aligned}\Delta V &= \frac{1}{2}(y_k^T Qy_k + 2y_k^T Su_k + u_k^T Ru_k) \\ &- \frac{1}{2}(x_k^T LL^T x_k + x_k^T LWu_k + u_k^T W^T L^T x_k + u_k^T W^T Wu_k)\end{aligned}$$

Summing over the time interval from $k = 0$ to $k = N$, based on the sampling time, T_s. The total energy equation becomes

$$\begin{aligned}\frac{T_s}{2}\sum_{k=0}^{N}(y_k^T Qy_k + 2y_k^T Su_k + u_k^T Ru_k) - T_s(V_{k+1} - V_{k_0}) \\ -\frac{T_s}{2}\sum_{k=0}^{N}(x_k^T LL^T x_k + x_k^T LWu_k + u_k^T W^T L^T x_k + u_k^T W^T Wu_k) = 0\end{aligned} \tag{129}$$

where

$$\begin{aligned}
E_{su} &= \frac{T_s}{2}\sum_{k=0}^{N}(y_k^T Q y_k + 2y_k^T S u_k + u_k^T R u_k)\\
E_{st} &= T_s(V_{k+1} - V_{k_0})\\
E_d &= \frac{T_s}{2}\sum_{k=0}^{N}(x_k^T L L^T x_k + x_k^T L W u_k + u_k^T W^T L^T x_k + u_k^T W^T W u_k)
\end{aligned}$$

Hence, $E_T = E_{su} - E_{st} - E_d = 0$ □

The system under consideration is a networked system, whereby the components of the system communicate over a packet-switched network. Hence, it is appropriate to directly relate the energy based equation to the transmitted and received components over the network. We now provide the energy balance in terms of the exchanged wave variables.

Proposition 7.4. *Given the system $\mathcal{H}_p$ with the energy balance as defined in (129) and the wave transformation provided in (107)-(110). The resulting energy balance of the system in wave domain is*

$$E_{T_{wv}} = E_{su_{wv}} - E_{st_{wv}} - E_{d_{wv}} = 0 \tag{130}$$

where $E_{T_{wv}}$ is the total of the system, $E_{su_{wv}}$ is the supplied energy, $E_{st_{wv}}$ is the stored energy and $E_{d_{wv}}$ is the dissipated energy.

Proof. From equations (9) and (10), and also assuming **b**$= 1$, solving for the plant output, y_k and input, u_k, we have

$$y_k = \frac{1}{\sqrt{2}}(U_{r_k} + V_{r_k}) \tag{131}$$

$$u_k = \frac{1}{\sqrt{2}}(U_{r_k} - V_{r_k}) \tag{132}$$

After some mathematical manipulations and simplification, the plant dynamics can be expressed in terms of the input wave variable, U_{r_k} and output wave variable, V_{r_k}. The resulting system, $\mathcal{H}_{p_{wv}}$ can be described as

$$\mathcal{H}_{p_{wv}} : \begin{cases} x_{k+1} = \bar{A}x_k + \bar{B}U_{r_k} \\ V_{r_k} = \bar{C}x_k + \bar{D}U_{r_k} \end{cases} \tag{133}$$

with

$$\begin{aligned}\bar{A} &= A - B(D+I)^{-1}C \\ \bar{B} &= \frac{B}{\sqrt{2}}(I - (D+I)^{-1}(D-I)) \\ \bar{C} &= \sqrt{2}(D+I)^{-1}C \\ \bar{D} &= (D+I)^{-1}(D-I)\end{aligned}$$

Recall the total energy expression given in (129), by substitution, the energy balance in the wave domain becomes

$$\begin{aligned}&\frac{T_s}{2}\sum_{k=0}^{N}(V_{r_k}^T\bar{Q}V_{r_k} + 2V_{r_k}^T\bar{S}U_{r_k} + U_{r_k}^T\bar{R}U_{r_k}) - T_s(V_{k+1} - V_0) \\ -&\frac{T_s}{2}\sum_{k=0}^{N}(x_k^T\overline{LL^T}x_k + x_k^T\overline{LW}U_{r_k} + U_{r_k}^T\overline{W^TL^T}x_k + U_{r_k}^T\overline{W^TW}U_{r_k}) = 0 \qquad (134)\end{aligned}$$

where

$$\bar{Q} = (\frac{Q-2S+R}{2}); \quad \bar{S} = (\frac{Q-R}{2}); \quad \bar{R} = (\frac{Q+2S+R}{2}); \qquad (135)$$

$$\begin{aligned}\overline{LL^T} &= (LL^T - \frac{LW\bar{C}}{\sqrt{2}} - \frac{\bar{C}W^TL^T}{\sqrt{2}} + \bar{C}^TW^TW\bar{C}) \\ \overline{LW} &= (\frac{LW}{\sqrt{2}} - \frac{LW\bar{D}}{\sqrt{2}} - \frac{\bar{C}^TW^TW}{2} + \frac{\bar{C}^TW^TW\bar{D}}{2}) \\ \overline{W^TL^T} &= (\frac{W^TL^T}{\sqrt{2}} - \frac{\bar{D}^TW^TL^T}{\sqrt{2}} - \frac{W^TW\bar{C}}{2} + \frac{\bar{D}^TW^TW\bar{C}}{2}) \\ \overline{W^TW} &= (\frac{W^TW - W^TW\bar{D} - \bar{D}W^TW + \bar{D}^TW^TW\bar{D}}{2})\end{aligned}$$

With

$$\begin{aligned}
E_{su_{wv}} &= \frac{T_s}{2}\sum_{k=0}^{N}(V_{r_k}^T\bar{Q}V_{r_k} + 2V_{r_k}^T\bar{S}U_{r_k} + U_{r_k}^T\bar{R}U_{r_k}) \\
E_{st_{wv}} &= T_s(V_{k+1} - V_0) \\
E_{d_{wv}} &= \frac{T_s}{2}\sum_{k=0}^{N}(x_k^T\overline{LL^T}x_k + x_k^T\overline{LW}U_{r_k} + U_{r_k}^T\overline{W^TL^T}x_k + U_{r_k}^T\overline{W^TW}U_{r_k})
\end{aligned}$$

Hence, $E_{T_{wv}} = E_{su_{wv}} - E_{st_{wv}} - E_{d_{wv}} = 0$ □

7.3.2 Main Results: Characterization of Energy in the Presence of Attacks

In this section, we provide a generalized characterization of the total energy in the presence of attacks.

Theorem 7.5. *Consider the networked control system depicted in Figure 66, under cyber-attack, $\mathcal{A}_k$, where by the attacker can remove or modify the exchanged wave variables U_{r_k} and V_{r_k}. Since the plant is assumed to be linear and time-invariant, the modified variables due to an attack can be modeled as*

$$\tilde{U}_{r_k} = U_{r_k} + U_{a_k}; \qquad \tilde{V}_{r_k} = V_{r_k} + V_{a_k}; \qquad \tilde{x}_k = x_k + x_{a_k}; \tag{136}$$

The total energy of the plant system, $\tilde{E}_{T_{wv}}$, in the presence of attack is

$$\tilde{E}_{T_{wv}} = E_{T_a} \neq 0 \tag{137}$$

Proof. Based on the attack-modified input-output relations, the energy for the system becomes

$$\begin{aligned}
\tilde{E}_{T_{wv}} &= \frac{T_s}{2}\sum_{k=0}^{N}(\tilde{V}_{r_k}^T\bar{Q}\tilde{V}_{r_k} + 2\tilde{V}_{r_k}^T\bar{S}\tilde{U}_{r_k} + \tilde{U}_{r_k}^T\bar{R}\tilde{U}_{r_k}) \\
&- \frac{T_s}{2}\sum_{k=0}^{N}(\tilde{x}_k\overline{LL^T}\tilde{x}_k + \tilde{x}_k\overline{LW}\tilde{U}_{r_k} + \tilde{U}_{r_k}\overline{W^TL^T}\tilde{x}_k + \tilde{U}_{r_k}\overline{W^TW}\tilde{U}_{r_k}) \\
&- T_s(\frac{1}{2}\tilde{x}_{k+1}P\tilde{x}_{k+1} - \frac{1}{2}\tilde{x}_0P\tilde{x}_0)
\end{aligned} \tag{138}$$

Next, we simplify the above total energy based on the individual energy components which include supplied, stored and dissipated energies. For the new supplied energy we have,

$$\tilde{E}_{su_{wv}} = \frac{T_s}{2}\sum_{k=0}^{N}(\tilde{V}_{r_k}^T\bar{Q}\tilde{V}_{r_k} + 2\tilde{V}_{r_k}^T\bar{S}\tilde{U}_{r_k} + \tilde{U}_{r_k}^T\bar{R}\tilde{U}_{r_k}) \tag{139}$$

substituting (136) in (139), we have

$$\begin{aligned}\tilde{E}_{su_{wv}} &= \frac{T_s}{2}\sum_{k=0}^{N}((V_{r_k}+V_{a_k})^T\bar{Q}(V_{r_k}+V_{a_k}) + 2(V_{r_k}+V_{a_k})^T\bar{S}(U_{r_k}+U_{a_k})\\ &+ (U_{r_k}+U_{a_k})^T\bar{R}(U_{r_k}+U_{a_k}))\\ &= \frac{T_s}{2}\sum_{k=0}^{N}(V_{r_k}^T\bar{Q}V_{r_k} + 2V_{r_k}^T\bar{Q}V_{a_k} + V_{a_k}^T\bar{Q}V_{a_k} + 2V_{r_k}^T\bar{S}U_{r_k} + 2V_{r_k}^T\bar{S}U_{a_k}\\ &+ 2V_{a_k}^T\bar{S}U_{r_k} + 2V_{a_k}^T\bar{S}U_{a_k} + U_{r_k}^T\bar{R}U_{r_k} + 2U_{r_k}^T\bar{R}U_{a_k} + U_{a_k}^T\bar{R}U_{a_k})\\ &= \frac{T_s}{2}\sum_{k=0}^{N}(V_{r_k}^T\bar{Q}V_{r_k} + 2V_{r_k}^T\bar{S}U_{r_k} + U_{r_k}^T\bar{R}U_{r_k})\\ &+ \frac{T_s}{2}\sum_{k=0}^{N}(V_{a_k}^T\bar{Q}V_{a_k} + 2V_{r_k}^T\bar{Q}V_{a_k} + 2V_{r_k}^T\bar{S}U_{a_k})\\ &+ \frac{T_s}{2}\sum_{k=0}^{N}(2V_{a_k}^T\bar{S}U_{r_k} + 2V_{a_k}^T\bar{S}U_{a_k} + 2U_{r_k}^T\bar{R}U_{a_k} + U_{a_k}^T\bar{R}U_{a_k})\end{aligned} \tag{140}$$

From (140) above, it can be seen that,

$$\tilde{E}_{su_{wv}} = E_{su_{wv}} + E_{su_{wva}} \tag{141}$$

Next, the new stored energy component becomes,

$$\tilde{E}_{st_{wv}} = T_s(\frac{1}{2}\tilde{x}_{k+1}^T P\tilde{x}_{k+1} - \frac{1}{2}\tilde{x}_0^T P\tilde{x}_0) \tag{142}$$

substituting (136) in (142), we have

$$\begin{aligned}\tilde{E}_{st_{wv}} &= T_s(\frac{1}{2}(x_{k+1} + x_{a_{k+1}})^T P(x_{k+1} + x_{a_{k+1}}) - \frac{1}{2}(x_0 + x_{a_0})^T P(x_0 + x_{a_0}))\\ &= \frac{T_s}{2}((x_{k+1}^T P x_{k+1} + 2x_{k+1}^T P x_{a_{k+1}} + x_{a_{k+1}}^T P x_{a_{k+1}}) - (x_0^T P x_0 + 2x_0^T P x_{a_0} + x_{a_0}^T P x_{a_0}))\\ &= \frac{T_s}{2}((x_{k+1}^T P x_{k+1} - x_0^T P x_0) + (x_{a_{k+1}}^T P x_{a_{k+1}} - x_{a_0}^T P x_{a_0} + 2x_{k+1}^T P x_{a_{k+1}} - x_0^T P x_{a_0}))\end{aligned} \tag{143}$$

From (143) above, it can be seen that,

$$\tilde{E}_{st_{wv}} = E_{st_{wv}} + E_{st_{wva}} \tag{144}$$

Finally, the new dissipated energy component becomes

$$\tilde{E}_{d_{wv}} = \frac{T_s}{2}\sum_{k=0}^{N}(\tilde{x}_k \overline{LL^T} \tilde{x}_k + \tilde{x}_k^T \overline{LW} \tilde{U}_{rk} + \tilde{U}_{rk} \overline{W^T L^T} \tilde{x}_k + \tilde{U}_{rk} \overline{W^T W} \tilde{U}_{rk}) \tag{145}$$

substituting (136) in (145), we have

$$\begin{aligned}\tilde{E}_{d_{wv}} &= \frac{T_s}{2}\sum_{k=0}^{N}((x_k + \overline{x}_k)^T \overline{LL^T}(x_k + \overline{x}_k) + (x_k + \overline{x}_k)\overline{LW}(U_{rk} + U_{a_k})\\ &+ (U_{rk} + U_{a_k})\overline{W^T L^T}(x_k + \overline{x}_k) + (U_{rk} + U_{a_k})\overline{W^T W}(U_{rk} + U_{a_k}))\\ &= \frac{T_s}{2}\sum_{k=0}^{N}(x_k^T \overline{LL^T} x_k + x_k^T \overline{LL^T} x_{a_k} + x_{a_k}^T \overline{LL^T} x_k + x_{a_k}^T \overline{LL^T} x_{a_k}\\ &+ x_k^T \overline{LW} U_{rk} + x_k^T \overline{LW} U_{a_k} + x_{a_k}^T \overline{LW} U_{rk} + x_{a_k}^T \overline{LW} U_{a_k}\\ &+ U_{rk}^T \overline{W^T L^T} x_k + U_{rk}^T \overline{W^T L^T} x_{a_k} + U_{a_k}^T \overline{W^T L^T} x_k + U_{a_k}^T \overline{W^T L^T} x_{a_k}\\ &+ U_{rk} \overline{W^T W} U_{rk} + U_{rk} \overline{W^T W} U_{a_k} + U_{a_k} \overline{W^T W} U_{rk} + U_{a_k} \overline{W^T W} U_{a_k})\end{aligned} \tag{146}$$

From (146) above, it can be seen that,

$$\tilde{E}_{d_{wv}} = E_{d_{wv}} + E_{d_{wva}} \tag{147}$$

Hence, from (141), (144) and (147), the total energy, $\tilde{E}_T$ in the presence of attack(s), then becomes

$$\begin{aligned}
\tilde{E}_{T_{wv}} &= \tilde{E}_{su_{wv}} - \tilde{E}_{st_{wv}} - \tilde{E}_{d_{wv}} \\
&= E_{su_{wv}} + E_{su_{wva}} - E_{st_{wv}} - E_{st_{wva}} - E_{d_{wv}} - E_{d_{wva}} \\
&=_{su_{wv}} -E_{st_{wv}} - E_{d_{wv}} +_{su_{wva}} -E_{st_{wva}} - E_{d_{wva}} \\
&= E_{T_{wv}} + E_{T_a} \qquad (148)
\end{aligned}$$

From (130), we have

$$\tilde{E}_T = E_{T_{wv}} + E_{T_a} = E_{T_a} \qquad (149)$$

□

Corollary 7.6. *In the absence of any detectable attack, $\mathcal{A}_k$, the total energy of the system, $\tilde{E}_{T_{wv}}$ is*

$$\tilde{E}_{T_{wv}} = E_{T_{wv}} = 0 \qquad (150)$$

Proof. This result follows directly from the system total energy property described in Theorem 7.4 and the results in Theorem 7.5 in the presence attacks. □

Remark 7.1. *The detection algorithm for the monitor is evaluated based on the information received at the controller. Considering the fact that the controller is considered trustworthy, the effects of attacks on the wave variable, U_{l_k} which is received as U_{r_k} by the plant will be reflected on wave variable V_{l_k}, which is V_{r_k} sent from the plant side of the network. Recall the expression in (108) relating the actuator signal and sensor signal , to the wave variable,*

$$V_{r_k} = \frac{1}{\sqrt{2b}}(y_k - bu_k)$$

It is straight forward to see that attacks on either the sensor or actuator will be reflected on the wave variable V_{r_k}, which is subsequently received at the controller as V_{l_k}.

Corollary 7.7. *An attack, $\mathcal{A}_k$, is characterized as a passive attack if the presence of the attack results in $\tilde{E}_{T_{wv}} > 0$.*

From the definition of passivity in (3.6), $\tilde{E}_{T_{wv}} > 0$ implies that the supplied energy for the attack system is larger than the dissipated and stored energies.

Corollary 7.8. *An attack, $\mathcal{A}_k$, is characterized as a non-passive attack if the presence of the attack results in $\tilde{E}_{T_{wv}} < 0$.*

This essentially implies that the supplied energy of the attacked system is less than the dissipated and store energies. Therefore, the system generates additional internal energy which results in a non-passive behavior. This implies that the overall stability of the networked control system is no longer guaranteed.

The energy-based attack detector can be summarized by Algorithm 1. Figure 67 also shows the block diagram for the energy based monitor. The inputs to the algorithm, also denoted in Figure 67, are the wave variables, V_{r_k} and U_{r_k} and the plant's state x_k. The output of the algorithm is Ψ, which provides information on whether an attack has occurred and the impact of the attack on the overall system. The blocks supplied energy, stored energy and dissipated energy in Figure 67 corresponds to the computation of the supplied energy, stored energy and dissipated energy respectively as indicated by lines 1-4 in Algorithm 1. Figure 67 shows the block diagram for the designed energy

Algorithm 1: Energy-Based Attack Detection

```
   Input: $V_{r_k}$,$U_{r_k}$,$x_k$
   Output: $\Psi$
1  Compute the supplied energy, $E_{su_{wv}}$
2  Compute the stored energy, $E_{st_{wv}}$
3  Compute the dissipated energy, $E_{d_{wv}}$
4  Compute the total energy, $E_{T_k} = E_{su_{wv}} - E_{st_{wv}} - E_{d_{wv}}$
5  if $E_{T_k} \neq 0$ then
6      $\psi_1$ =True
7      if $E_{T_k} > 0$ then
8          $\psi_2$ =Passive
9      else
10         $\psi_2$ =Non-Passive
11 else
12     $\psi_1$ =False
13 $\Psi$ =$\{\psi_1,\psi_2\}$
14 return $\Psi$
```

based monitor for attacks in the case of measurable plant states.

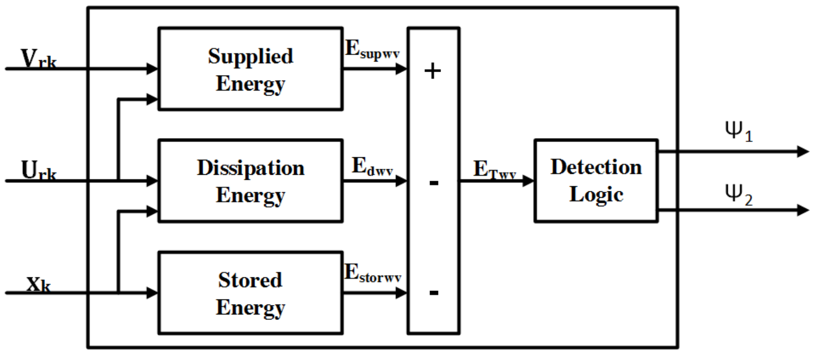


Figure 67: Energy-Based Monitor

Remark 7.2. *Thus far, the characterization of attacks are based on the notion that in the absence of attacks, the nominal energy balance of the monitored system should be zero. In a more realistic setting, this assumption can be relaxed in order to integrate the potential effects of the network communication as a result of delays or packet loss. In the presence of network effects, instead of the energy balance being zero, a notion of a maximal value of energy due to network effects is considered. Based on this notion, a threshold boundary, E_{th}, is defined. The characterization of attacks are then evaluated based on the impact of the attacks that results in computed energy that lies outside the boundary. This maximal energy value can be obtained empirically through various approaches such as simulations and by imposing worst-case network conditions for the NCS.*

7.3.3 Characterization of Attack Models

In order to illustrate the impact of attack models on the physical system, we evaluate the effect of classical attacks on the total energy of the system. For brevity, we focus on the attacks **A1** and **A2**, although similar approach can be used to evaluate the effects of attacks **A3**-**A6**. Also, we consider the cases where the dissipative plant is passive or strictly-ouput passive.

Assuming there are no attacks on U_{r_k}, the impact of the attacks on V_{r_k} is reflected on only the supplied energy resulting in the component,

$$\tilde{E}_{T_{wv}} = E_{T_{wva}} = E_{su_{wva}} = \frac{T_s}{2}\sum_{k=0}^{N}(2V_{r_k}^T\bar{Q}V_{a_k} + V_{a_k}^T\bar{Q}V_{a_k} + 2V_{a_k}^T\bar{S}U_{r_k}) \tag{151}$$

Proposition 7.9. *Consider the passivity-based network control system depicted in Figure 66, under* *a max integrity attack, $\mathcal{A}_k$, if the system dynamics $\mathcal{H}_p$ is passive, then*

$$\mathcal{A}_k : \begin{cases} \textit{Passive} & \textit{if} \quad \sum_{k=0}^{N} V_{r_{max}}^T V_{r_{max}} < \sum_{k=0}^{N} V_{r_k}^T V_{r_k} \\ \textit{Non-Passive} & \textit{if} \quad \sum_{k=0}^{N} V_{r_{max}}^T V_{r_{max}} > \sum_{k=0}^{N} V_{r_k}^T V_{r_k} \end{cases} \tag{152}$$

if the system dynamics $\mathcal{H}_p$ is strictly output passive, then

$$\mathcal{A}_k : \begin{cases} \textit{Passive} & \textit{if} \quad \sum_{k=0}^{N} (V_{r_{max}}^T V_{r_{max}} + \frac{2\epsilon}{(\epsilon+1)} V_{r_{max}}^T U_{r_k}) < \sum_{k=0}^{N} (V_{r_k}^T V_{r_k} + \frac{2\epsilon}{(\epsilon+1)} V_{r_k}^T U_{r_k}) \\ \textit{Non-Passive} & \textit{if} \quad \sum_{k=0}^{N} (V_{r_{max}}^T V_{r_{max}} + \frac{2\epsilon}{(\epsilon+1)} V_{r_{max}}^T U_{r_k}) > \sum_{k=0}^{N} (V_{r_k}^T V_{r_k} + \frac{2\epsilon}{(\epsilon+1)} V_{r_k}^T U_{r_k}) \end{cases} \tag{153}$$

Proof. Based on the max integrity attack model in (112), $V_{a_k} = V_{r_{max}} - V_{r_k}$. Substituting for V_{a_k} in (151), we have

$$\begin{aligned} E_{su_{wva}} &= \frac{T_s}{2} \sum_{k=0}^{N} (2V_{r_k}^T \bar{Q}(V_{r_{max}} - V_{r_k}) + (V_{r_{max}} - V_{r_k})^T \bar{Q}(V_{r_{max}} - V_{r_k}) \\ &+ 2(V_{r_{max}} - V_{r_k})^T \bar{S} U_{r_k}) \\ &= \frac{T_s}{2} \sum_{k=0}^{N} (2V_{r_k}^T \bar{Q} V_{r_{max}} - 2V_{r_k}^T \bar{Q} V_{r_k} + V_{r_{max}}^T \bar{Q} V_{r_{max}} - 2V_{r_k}^T \bar{Q} V_{r_{max}} + V_{r_k}^T \bar{Q} V_{r_k} \\ &+ 2V_{r_{max}} \bar{S} U_{r_k} - 2V_{r_k}^T \bar{S} U_{r_k}) \\ &= \frac{T_s}{2} \sum_{k=0}^{N} (V_{r_{max}}^T \bar{Q} V_{r_{max}} - V_{r_k}^T \bar{Q} V_{r_k} + 2V_{r_{max}} \bar{S} U_{r_k} - 2V_{r_k}^T \bar{S} U_{r_k}) \end{aligned} \tag{154}$$

1. **Passive physical system**: In Section (3.7), a dissipative system is passive if it has following QSR parameters, $Q = 0$, $S = \frac{1}{2}$ and $R = 0$. Substituting the QSR parameters in (135),

$$\bar{Q} = \frac{Q - 2S + R}{2} = -\frac{1}{2} \tag{155}$$

$$\bar{S} = \frac{Q - R}{2} = 0 \tag{156}$$

Substituting the (155) and (156) in (154), we have

$$E_{su_{wva}} = -\frac{T_s}{4}\sum_{k=0}^{N}(V_{r_{max}}^T V_{r_{max}} - V_{r_k}^T V_{r_k}) \tag{157}$$

From (157), a max integrity attack is categorized as a passive attack if $E_{su_{wva}} > 0$ and as non-passive attack if $E_{su_{wva}} < 0$ which provides the result in (152)

2. **Strictly output passive physical system**: In Section (3.7), strictly-output passivity is equivalent to having the following QSR parameters, $Q = -\epsilon I$, $S = \frac{1}{2}$ and $R = 0$. Substituting the QSR parameters in (135),

$$\bar{Q} = \frac{Q - 2S + R}{2} = -\frac{\epsilon + 1}{2} \tag{158}$$

$$\bar{S} = \frac{Q - R}{2} = -\frac{\epsilon}{2} \tag{159}$$

Substituting (158) and (159) in in (154), we have

$$E_{su_{wva}} = -\frac{T_s}{2}\sum_{k=0}^{N}(\frac{(\epsilon+1)V_{r_{max}}^T V_{r_{max}}}{2} - \frac{(\epsilon+1)V_{r_k}^T V_{r_k}}{2} + \epsilon V_{r_{max}}^T U_{r_k} - \epsilon V_{r_k}^T U_{r_k})$$

After further simplification, we have

$$E_{su_{wva}} = -\frac{T_s(\epsilon+1)}{4}\sum_{k=0}^{N}((V_{r_{max}}^T V_{r_{max}} + \frac{2\epsilon}{(\epsilon+1)}V_{r_{max}}^T U_{r_k}) - (V_{r_k}^T V_{r_k} + \frac{2\epsilon}{(\epsilon+1)}V_{r_k}^T U_{r_k})) \tag{160}$$

From (160), a max integrity attack is categorized as a passive attack if $E_{su_{wva}} > 0$ and as a non-passive attack if $E_{su_{wva}} < 0$ leading to the result in (153)

□

Proposition 7.10. *Consider the passivity-based network control system depicted in Figure 66, under* *a min integrity attack, $\mathcal{A}_k$, if the system dynamics $\mathcal{H}_p$ is passive, then*

$$\mathcal{A}_k : \begin{cases} \textit{Passive} & \textit{if} \quad \sum_{k=0}^{N} V_{r_{min}}^T V_{r_{min}} < \sum_{k=0}^{N} V_{r_k}^T V_{r_k} \\ \textit{Non-Passive} & \textit{if} \quad \sum_{k=0}^{N} V_{r_{min}}^T V_{r_{min}} > \sum_{k=0}^{N} V_{r_k}^T V_{r_k} \end{cases} \tag{161}$$

if the system dynamics $\mathcal{H}_p$ is strictly output passive, then

$$\mathcal{A}_k : \begin{cases} \textit{Passive} & \textit{if} \quad \sum_{k=0}^{N} (V_{r_{min}}^T V_{r_{min}} + \frac{2\epsilon}{(\epsilon+1)} V_{r_{min}}^T U_{r_k}) < \sum_{k=0}^{N} (V_{r_k}^T V_{r_k} + \frac{2\epsilon}{(\epsilon+1)} V_{r_k}^T U_{r_k}) \\ \textit{Non-Passive} & \textit{if} \quad \sum_{k=0}^{N} (V_{r_{min}}^T V_{r_{min}} + \frac{2\epsilon}{(\epsilon+1)} V_{r_{min}}^T U_{r_k}) > \sum_{k=0}^{N} (V_{r_k}^T V_{r_k} + \frac{2\epsilon}{(\epsilon+1)} V_{r_k}^T U_{r_k}) \end{cases} \tag{162}$$

Proof. Based on the min integrity attack model in (111), $V_{a_k} = V_{r_{min}} - V_{r_k}$. Substituting for V_{a_k} in (151), we have

$$\begin{aligned} E_{su_{wva}} &= \frac{T_s}{2} \sum_{k=0}^{N} (2V_{r_k}^T \bar{Q}(V_{r_{min}} - V_{r_k}) + (V_{r_{min}} - V_{r_k})^T \bar{Q}(V_{r_{min}} - V_{r_k}) \\ &+ 2(V_{r_{min}} - V_{r_k})^T \bar{S} U_{r_k}) \\ &= \frac{T_s}{2} \sum_{k=0}^{N} (2V_{r_k}^T \bar{Q} V_{r_{min}} - 2V_{r_k}^T \bar{Q} V_{r_k} + V_{r_{min}}^T \bar{Q} V_{r_{min}} - 2V_{r_k}^T \bar{Q} V_{r_{min}} + V_{r_k}^T \bar{Q} V_{r_k} \\ &+ 2V_{r_{min}} \bar{S} U_{r_k} - 2V_{r_k}^T \bar{S} U_{r_k}) \\ &= \frac{T_s}{2} \sum_{k=0}^{N} (V_{r_{min}}^T \bar{Q} V_{r_{min}} - V_{r_k}^T \bar{Q} V_{r_k} + 2V_{r_{min}} \bar{S} U_{r_k} - 2V_{r_k}^T \bar{S} U_{r_k}) \end{aligned} \tag{163}$$

1. **Passive physical system**: Substituting the (155) and (156) in (163), we have

$$E_{su_{wva}} = -\frac{T_s}{4} \sum_{k=0}^{N} (V_{r_{min}}^T V_{r_{min}} - V_{r_k}^T V_{r_k}) \tag{164}$$

   From (164), a min integrity attack is categorized as a passive attack if $E_{su_{wva}} > 0$ and as a non-passive attack if $E_{su_{wva}} < 0$ essentially leading to the result in (161)

2. **Strictly output passive physical system**: Substituting (158) and (159) in in (163), we have

$$E_{su_{wva}} = -\frac{T_s}{2}\sum_{k=0}^{N}(\frac{(\epsilon+1)V_{r_{min}}^T V_{r_{min}}}{2} - \frac{(\epsilon+1)V_{r_k}^T V_{r_k}}{2} + \epsilon V_{r_{min}}^T U_{r_k} - \epsilon V_{r_k}^T U_{r_k})$$

After further simplification, we have

$$E_{su_{wva}} = -\frac{T_s(\epsilon+1)}{4}\sum_{k=0}^{N}((V_{r_{min}}^T V_{r_{min}} + \frac{2\epsilon}{(\epsilon+1)}V_{r_{min}}^T U_{r_k}) - (V_{r_k}^T V_{r_k} + \frac{2\epsilon}{(\epsilon+1)}V_{r_k}^T U_{r_k})) \tag{165}$$

From (165), a min integrity attack is a categorized as a passive attack if $E_{su_{wva}} > 0$ and as a non-passive attack if $E_{su_{wva}} < 0$ essentially leading to the result in (162)

□

Proposition 7.11. *Consider the passivity-based network control system depicted in Figure 66, under an additive integrity attack, $\mathcal{A}_k$, if the system dynamics $\mathcal{H}_p$ is passive, then*

$$\mathcal{A}_k : \begin{cases} \textit{Passive} & \textit{if} \quad \sum_{k=0}^{N} 2V_{r_k}^T\alpha_k < -\sum_{k=0}^{N}\alpha_k^T\alpha_k \\ \textit{Non-Passive} & \textit{if} \quad \sum_{k=0}^{N} 2V_{r_k}^T\alpha_k > -\sum_{k=0}^{N}\alpha_k^T\alpha_k \end{cases} \tag{166}$$

if the system dynamics $\mathcal{H}_p$ is strictly output passive, then

$$\mathcal{A}_k : \begin{cases} \textit{Passive} & \textit{if} \quad \sum_{k=0}^{N} 2V_{r_k}^T\alpha_k + \frac{2\epsilon}{(\epsilon+1)}U_{r_k}^T\alpha_k < -\sum_{k=0}^{N}\alpha_k^T\alpha_k \\ \textit{Non-Passive} & \textit{if} \quad \sum_{k=0}^{N} 2V_{r_k}^T\alpha_k + \frac{2\epsilon}{(\epsilon+1)}U_{r_k}^T\alpha_k > -\sum_{k=0}^{N}\alpha_k^T\alpha_k \end{cases} \tag{167}$$

Proof. Based on the additive integrity attack model in (119), $V_{a_k} = \alpha_k$. Substituting for V_{a_k} in (151), we have

$$E_{su_{wva}} = \frac{T_s}{2}\sum_{k=0}^{N}(2V_{r_k}^T\bar{Q}\alpha_k + \alpha_k^T\bar{Q}\alpha_k + 2\alpha_k^T\bar{S}U_{r_k}) \tag{168}$$

1. **Passive physical system**: Substituting the (155) and (156) in (168), we have

$$E_{su_{wva}} = -\frac{T_s}{4}\sum_{k=0}^{N}(2V_{r_k}^T\alpha_k + \alpha_k^T\alpha_k) \tag{169}$$

From (169), an additive integrity attack is a categorized as a passive attack if $E_{su_{wva}} > 0$ and as non-passive attack if $E_{su_{wva}} < 0$ leading to the result in (166)

2. **Strictly output passive physical system**: Substituting (158) and (159) in in (168), we have

$$E_{su_{wva}} = -\frac{T_s}{2}\sum_{k=0}^{N}(2\frac{(\epsilon+1)V_{r_k}^T\alpha_k}{2} + \frac{(\epsilon+1)\alpha_k^T\alpha_k}{2} + \epsilon\alpha_k^T U_{r_k})$$

$$E_{su_{wva}} = -\frac{T_s(\epsilon+1)}{4}\sum_{k=0}^{N}((2V_{r_k}^T\alpha_k + \alpha_k^T\alpha_k + \frac{2\epsilon}{(\epsilon+1)}U_{r_k}^T\alpha_k)) \tag{170}$$

From (170), an additive integrity attack is categorized as a passive attack if $E_{su_{wva}} > 0$ and as a non-passive attack if $E_{su_{wva}} < 0$ essentially leading to the result in (167)

□

Proposition 7.12. *Consider the passivity-based network control system depicted in Figure 66, under a denial-of-service attack, $\mathcal{A}_k$, if the system dynamics $\mathcal{H}_p$ is passive, then*

$$\mathcal{A}_k : \begin{cases} \textit{Passive} \quad \textit{with} \quad E_{su_{wva}} = \frac{T_s}{2}\sum\limits_{k=0}^{N}\frac{V_{r_k}^T V_{r_k}}{2} > 0 \quad \forall V_{r_k} \neq 0 \end{cases} \tag{171}$$

if the system dynamics $\mathcal{H}_p$ is strictly output passive, then

$$\mathcal{A}_k : \begin{cases} \textit{Passive} \quad \textit{with} \quad E_{su_{wva}} = \frac{T_s}{2}\sum_{k=0}^{N}(\frac{(\epsilon+1)V_{r_k}^T V_{r_k}}{2} + \epsilon V_{r_k}^T U_{r_k}) \end{cases} \tag{172}$$

Proof. Based on the DoS attack model in (123), $V_{a_k} = -V_{r_k}$. Substituting for V_{a_k} in (151), we have

$$E_{su_{wva}} = \frac{T_s}{2}\sum_{k=0}^{N}(2V_{r_k}^T\bar{Q}(-V_{r_k}) + (-V_{r_k})^T\bar{Q}(-V_{r_k}) + 2(-V_{r_k})^T\bar{S}U_{r_k})$$
$$E_{su_{wva}} = \frac{T_s}{2}\sum_{k=0}^{N}(V_{r_k}^T\bar{Q}V_{r_k} - 2V_{r_k}^T\bar{Q}V_{r_k} - 2V_{r_k}^T\bar{S}U_{r_k}) = -\frac{T_s}{2}\sum_{k=0}^{N}(V_{r_k}^T\bar{Q}V_{r_k} + 2V_{r_k}^T\bar{S}U_{r_k}) \tag{173}$$

1. **Passive physical system**: Substituting the (155) and (156) in (173), we have

$$E_{su_{wva}} = \frac{T_s}{2}\sum_{k=0}^{N}\frac{V_{r_k}^T V_{r_k}}{2} \tag{174}$$

   From (174), one can see that $E_{su_{wva}}$ is always greater than zero for $V_{r_k} \neq 0$ hence a DoS attack is always a passive attack.

2. **Strictly output passive physical system**: Substituting (158) and (159) in in (173), we have

$$E_{su_{wva}} = \frac{T_s}{2}\sum_{k=0}^{N}(\frac{(\epsilon+1)V_{r_k}^T V_{r_k}}{2} + \epsilon V_{r_k}^T U_{r_k}) \tag{175}$$

   By definition of passivity $V_{r_k}^T U_{r_k} \geq 0$ and given that $\epsilon > 0$, then $E_{su_{wva}} > 0$. Similar to the passive case, from (175), a DoS attack is always a passive attack

□

Remark 7.3. *The result obtained for the characterization of DoS attacks is similar for the analysis of packet losses due to unreliability of network in the literature. While packet losses are due to unreliable network, DoS is as a result of intentional and malicious attacks by an adversary.*

For the class of linear systems considered in the work, EBAD can appropriately detect all the attacks considered in Table 9.

7.3.4 Total Energy in the case of unmeasurable states

In the case unmeasurable plant states, a Luenberger observer of the form in (176) is introduced to reconstruct an estimate of the plant states.

$$\mathcal{H}_{obs}: \begin{cases} \hat{x}_{k+1} = A\hat{x}_k + BU_{r_k} - L(V_{r_k} - C\hat{x}_k - DU_{r_k}) \\ \hat{V}_{r_k} = C\hat{x}_k + DU_{r_k} \end{cases} \tag{176}$$

where L is the observer gain. Recall that the plant system is assumed to be observable. This means there exists an observability matrix L such that the estimated state $\hat{x}_k$ of the Luenberger observer asymptotically converges to the true state x_k.

Proposition 7.13. *Given the system $\mathcal{H}_p$ with the energy balance described in Theorem 7.4. In the case whereby the states are unmeasurable assuming a Luenberger observer, $\mathcal{H}_{obsv}$ as given in (176) is integrated to estimate the states, $\hat{x}_k$. Then, the resulting equivalent total energy of the system in wave domain, in the absence of attacks can be described by*

$$E_{T_{wv}} = E_{T_{wvo}} = -E_{st_{oe}} - E_{d_{oe}} \tag{177}$$

Proof. Due to the non-measurable states, the resulting energy-balance, $E_{T_{wv}}$ is modified. Let, the observation error be represented by $e_k = x_k - \hat{x}_k$ From the observation error, the estimated state is defined as $\hat{x}_k = x_k - e_k$. The total energy for the case of non-measurable states can be described by

$$E_{T_{wvo}} = E_{su_{wvo}} - E_{st_{wvo}} - E_{d_{wvo}} \tag{178}$$

$$\begin{aligned} \text{Where} \quad E_{su_{wvo}} &= E_{su_{wv}} \\ E_{st_{wvo}} &= T_s(\frac{1}{2}\hat{x}_{k+1}^T P\hat{x}_{k+1} - \frac{1}{2}\hat{x}_{k_0}P\hat{x}_{k_0}) \\ E_{d_{wvo}} &= \frac{T_s}{2}\sum_{k=0}^{N}(\hat{x}_k^T\overline{LL^T}\hat{x}_k + \hat{x}_k^T\overline{LW}u_k + U_{r_k}^T\overline{W^T L^T}\hat{x}_k + U_{r_k}^T\overline{W^T W}U_{r_k}) \end{aligned}$$

Substituting $\hat{x}_k = x_k - e_k$ into $E_{st_{wvo}}$ above, we have

$$\begin{aligned}
E_{st_{wvo}} &= T_s(\frac{1}{2}\hat{x}_{k+1}^T P\hat{x}_{k+1} - \frac{1}{2}\hat{x}_{k_0} P\hat{x}_{k_0}) \\
&= T_s(\frac{1}{2}(x_{k+1} - e_{k+1})^T P(x_{k+1} - e_{k+1}) - \frac{1}{2}(x_{k_0} - e_{k_0})^T P(x_{k_0} - e_{k_0})) \\
&= E_{st_{wv}} + \frac{T_s}{2}(e_{k+1}^T Pe_{k+1} - 2x_{k+1}^T Pe_{k+1} + 2x_{k_0}^T Px_{k_0} + e_{k_0}^T Pe_{k_0}) \\
E_{st_{wvo}} &= E_{st_{wv}} + E_{st_{oe}} \qquad (179)
\end{aligned}$$

Similarly, substituting $\hat{x}_k = x_k - e_k$ into $E_{d_{wvo}}$ above, we have

$$\begin{aligned}
E_{d_{wvo}} &= \frac{T_s}{2}\sum_{k=0}^{N}(\hat{x}_k^T\overline{LL^T}\hat{x}_k + \hat{x}_k^T\overline{LW}U_{r_k} + U_{r_k}^T\overline{W^TL^T}\hat{x}_k + U_{r_k}^T\overline{W^TW}U_{r_k}) \\
&= \frac{T_s}{2}\sum_{k=0}^{N}((x_k - e_k)^T\overline{LL^T}(x_k - e_k) + (x_k - e_k)^T\overline{LW}U_{r_k}) + U_{r_k})^T\overline{W^TL^T}(x_k - e_k) \\
&+ U_{r_k})^T\overline{W^TW}U_{r_k})) \\
&= \frac{T_s}{2}\sum_{k=0}^{N}(x_k^T\overline{LL^T}x_k - x_k^T\overline{LL^T}e_k - e_k^T\overline{LL^T}x_k + e_k^T\overline{LL^T}e_k + x_k^T\overline{LW}U_{r_k} - e_k^T\overline{LW}U_{r_k}
\end{aligned}$$

$$\begin{aligned}
&+ U_{r_k}^T\overline{W^TL^T}x_k - U_{r_k}^T\overline{W^TL^T}e_k + U_{r_k}^T\overline{W^TW}U_{r_k}) \\
&= \frac{T_s}{2}\sum_{k=0}^{N}(x_k^T\overline{LL^T}x_k + x_k^T\overline{LW}U_{r_k} + U_{r_k}^T\overline{W^TL^T}x_k + U_{r_k}^T\overline{W^TW}U_{r_k}) \\
&+ \frac{T_s}{2}\sum_{k=0}^{N}(-x_k^T\overline{LL^T}e_k - e_k^T\overline{LL^T}x_k + e_k^T\overline{LL^T}e_k - e_k^T\overline{LW}U_{r_k} - U_{r_k}^T\overline{W^TL^T}e_k) \\
E_{d_{wvo}} &= E_{d_{wv}} + E_{d_{oe}} \qquad (180)
\end{aligned}$$

Substituting (179) and (180) in (178) we have

$$\begin{aligned}
E_{T_{wvo}} &= E_{su_{wvo}} - E_{st_{wvo}} - E_{d_{wvo}} \\
&= E_{su_{wv}} - (E_{st_{wv}} + E_{st_{oe}}) - (E_{d_{wv}} + E_{d_{oe}}) \\
&= E_{su_{wv}} - E_{st_{wv}} - E_{d_{wv}} - E_{st_{oe}} - E_{d_{oe}} \qquad (181)
\end{aligned}$$

Substituting (130) in the $E_{T_{wvo}}$ above results in following

$$E_{T_{wvo}} = -E_{st_{oe}} - E_{d_{oe}} \tag{182}$$

□

Similar to Algorithm 1, in the case of unmeasurable states, the energy-based detection with the integration of an observer can be summarized by Algorithm 2 below. Figure 68 also shows the block diagram of the energy-based monitor in the case of unmeasurable plant states. The additional block for the observer in Figure 68 corresponds to the estimation of the states in Algorithm 2.

Algorithm 2: Energy-Based Attack Detection in the case of unmeasurable states

```
   Input: $V_{r_k}$,$U_{r_k}$
   Output: $\Psi$
1  Estimate the states, $\hat{x}_k$
2  Compute the supplied energy, $E_{su_{wvo}}$
3  Compute the stored energy, $E_{st_{wvo}}$
4  Compute the dissipated energy, $E_{d_{wvo}}$
5  Compute the total energy, $T_{wvo} = E_{su_{wvo}} - E_{st_{wvo}} - E_{d_{wvo}}$
6  if $E_{T_{wvo}} \neq 0$ then
7      $\psi_1$ =True
8      if $T_{wvo} > 0$ then
9          $\psi_2$ =Passive
10     else
11         $\psi_2$ =Non-Passive
12 else
13     $\psi_1$ =False
14 $\Psi$ ={$\psi_1$,$\psi_2$}
15 return $\Psi$
```

7.4 Simulation Results

In this section, we evaluate the proposed energy-based detection mechanism using simulations. The system under consideration is a control system composed of a plant and controller that exchange information over a network in order to cooperatively achieve a specified objective.

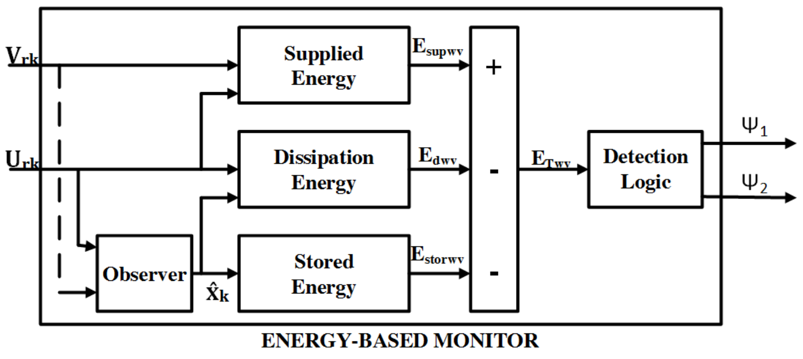


Figure 68: Energy-Based Monitor (Non-Measurable States)

7.4.1 Simulation Setup

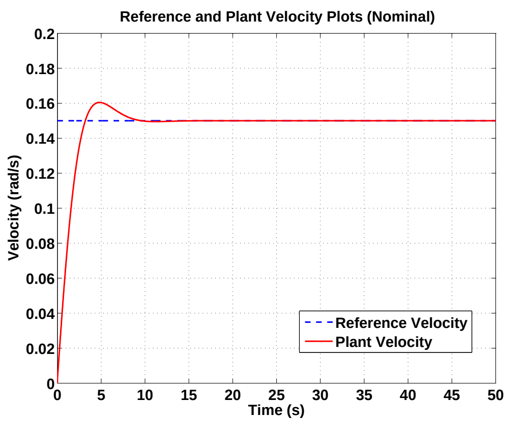
The case study involves the velocity control of a single joint robotic arm over a communication network. We briefly describe the model of the robotic joint and the networked controller. We then describe the attack detection component which is co-located with the networked controller. We present various attack scenarios to illustrate the performance of the proposed detection mechanism. It is assumed that the only information the networked controller receives from the plant is the wave variable, V_{r_k}, which becomes V_{l_k} at the controller side of the network. Hence the detection mechanism with an integrated observer, as shown in Figure 68, is used in this evaluation.

The simulation of the overall NCS is performed using Matlab/Simulink. The plant, controller, energy-based monitor, scattering transformation, attack models and communication are implemented using a combination of Matlab scripts and blocks from the Simulink library. The dynamics of the plant is described by the discrete-time state-space representation as defined in (105) with a sampling time of $T_s = 0.01s$. The parameters for the plant are A=0.9952, B=0.0625, C=0.1214, D=0.0251. The controller for the robot is a Proportional-Integral (PI) controller and similar to the plant is represented by the discrete time state space representation defined in (106) with the sampling time $T_s = 0.01s$. The parameters for the controller are A$_c$ =1, B$_c$ =0.0625, C$_c$ =0.1, D$_c$ =0.6385. The main objective of the controller is to modify the behavior of the plant in order to track a reference velocity trajectory, r_k over a communication network.

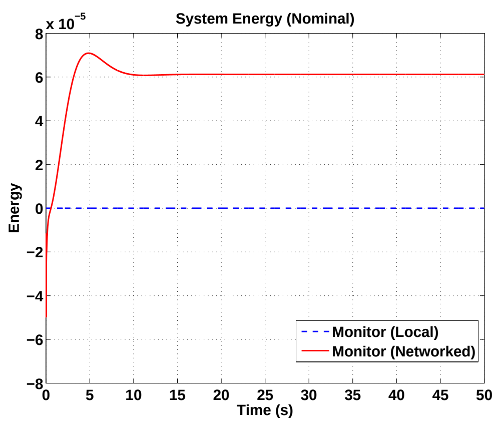
7.4.2 Scenarios

First, we present the control of the plant in the nominal case when there are no attacks. We also present the effects of the network on the system's energy. Next, we evaluate the behavior of the system under attack and the ability of the proposed approach to detect the attacks. In the experiments with attacks, the simulated attacks are injected from the duration, $t = 15s$ to $t = 20s$

1. *Nominal Case*: This evaluation considers the absence of attacks on the NCS. In this scenario, the NCS operates nominally while achieving the tracking objective. Figure 69a depicts the reference velocity of $0.15rad/s$ as well as the plant velocity clearly showing that the plant is able to track the velocity as desired. Figure 69b shows the energy balance of the monitored plant computed based on the approach described in Section 7.3.2. In order to illustrate the effect of communication network on the energy-balance, we also co-located the energy-based detector at the plant side of the network to essentially perform the same total energy computation. The only difference being the delay experience by the monitor co-located with the controller. From Figure 69b, the energy-balance computed by the local monitor is essentially zero as expected but the balance computed by the networked monitor has an offset as a result of the communication network. Hence, this offset value or a more conservative value can be used to characterize the threshold energy, E_{th}, which will be non-zero due to network effects.



(a) Velocity Plots.

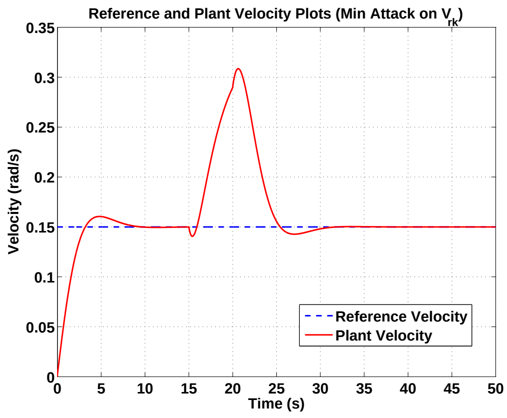


(b) Energy Balance Plot.

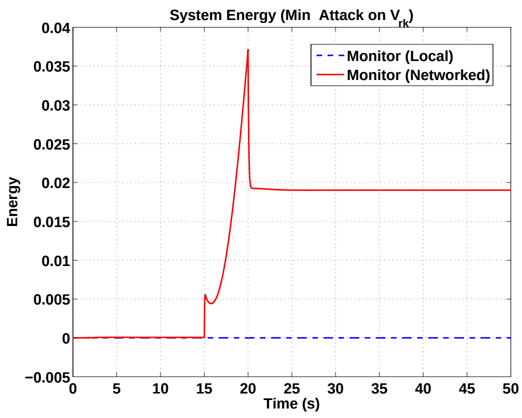
Figure 69: Simulation Results - Nominal Case.

2. *Integrity Attacks*

(a) *Min Attack on* V_{rk}: In this scenario, we introduce a min integrity attack on the NCS. We assume the NCS channel from the plant to the controller, transmitting V_{rk}, is compromised and during the attack duration, the attacker replaces the true or actual signal exchanged with the value of $V_{r_{min}}$. Figure 70a shows that the presence of the attack results in the degraded reference tracking performance. Figure 70b depicts the total energy computation clearly indicating the presence of the attack. Additionally, one can observe that the min integrity attack can be characterize as a passive attack as it leads to increase in the computed total energy which indicates the dissipation of energy. Based on the computed energy, passivity of the overall NCS is still guaranteed.



(a) Velocity Plots.

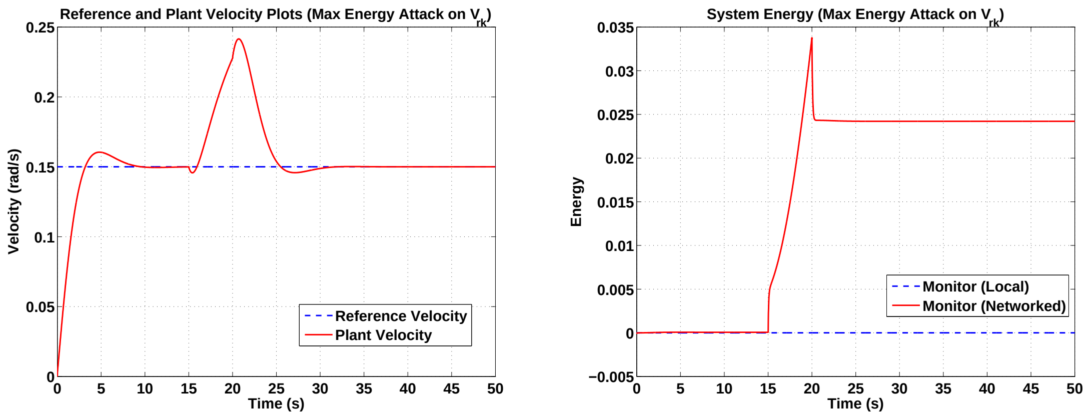


(b) Energy Balance Plot.

Figure 70: Simulation Results - Min Attack on V_{rk}.

(b) *Max Energy Attack on* V_{rk}: In this scenario, we introduce a max energy attack on the NCS. We assume the NCS channel from the plant to the controller, transmitting V_{rk}, is compromised and during the attack duration, the attacker replaces the true or actual signal exchanged with the value of V_{r_a} that maximizes the energy dissipated at that time step. Figure 71a shows that the presence of the attack results in the degraded reference tracking performance. Figure 71b depicts the total energy computation clearly indicating the presence of the attack. Additionally, one can observe that as expected the max energy attack results in an increase in the computed total energy. Based on the computed energy, passivity of the overall NCS is still guaranteed.

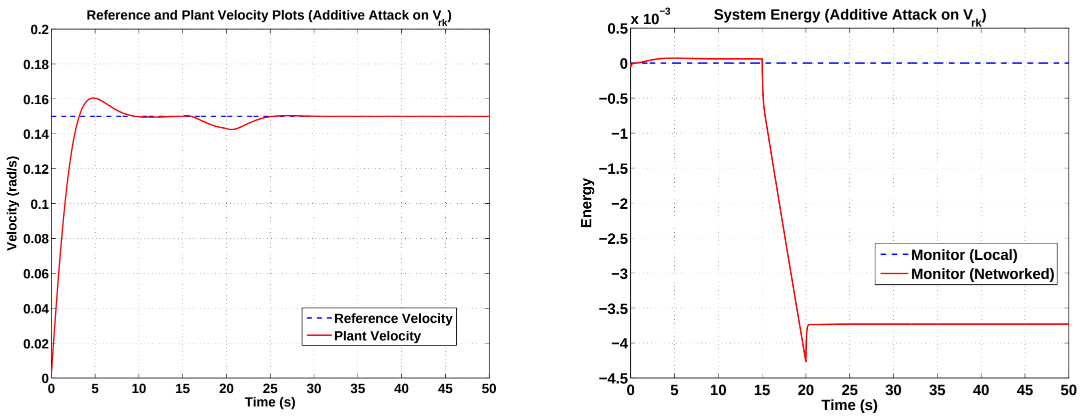
(c) *Additive Attack*: In this scenario, we introduce an additive integrity attack on the NCS. In this case, during the attack duration, the attacker introduces an additional bias, α to



(a) Velocity Plots. (b) Energy Balance Plot.

Figure 71: Simulation Results - Max Energy Attack on V_{rk}.

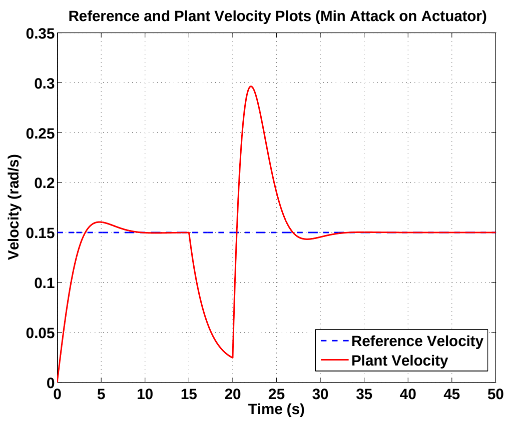
the true or actual signal exchanged. Figure 72a shows that the presence of the attack degrades the tracking performance slightly. Figure 72b depicts the total energy computation clearly indicating the presence of the attack. Additionally, one can observe that the additive integrity attack in this case can be characterize as a non-passive attack as it leads to a negative value in the computed total energy. In this case passivity of the overall system is no longer guaranteed although there's only a slight degradation of performance in tracking the desired reference.



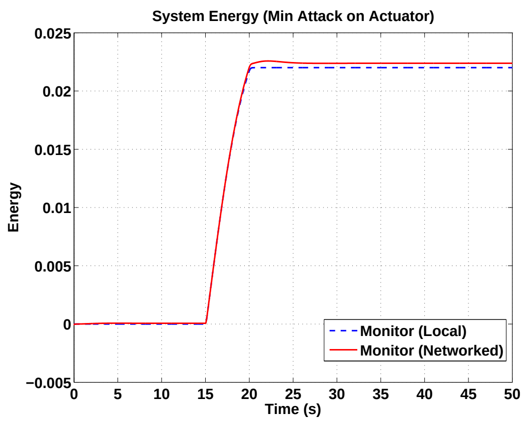
(a) Velocity Plots. (b) Energy Balance Plot.

Figure 72: Simulation Results - Additive Attack on V_{rk}.

(d) *Min Attack on the Actuator,* u_k: In this scenario, we introduce a min integrity attack on actuator of the plant. We assume the actuator command into the plant, u_k, is compromised and during the attack duration, the attacker replaces the true or actual signal exchanged with the value of u_{min}. Figure 73a shows that the presence of the attack results in the degraded reference tracking performance. Figure 73b depicts the total energy computation clearly indicating the presence of the attack. From the energy plots, one can observe that as expected the min integrity attack on the actuator perturbs both the local and networked energy monitors, clearly indicating that the attack is perpetuated locally. From Figure 73b, the min attack on the actuator results in an increase in the system energy and can be characterize as a passive attack. In this case the attack does not destroy passivity of the overall NCS but significantly affects the tracking performance.



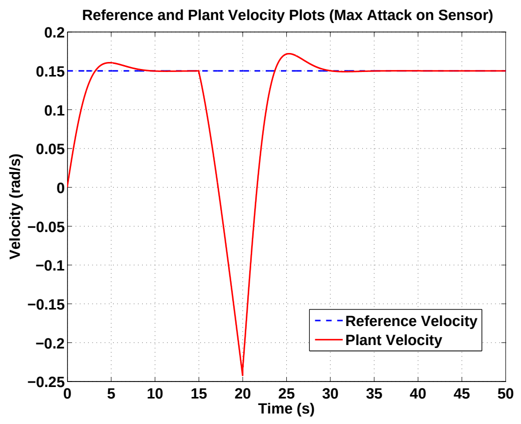
(a) Velocity Plots.



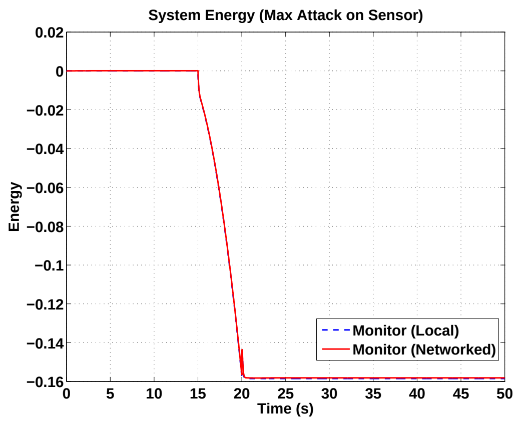
(b) Energy Balance Plot.

Figure 73: Simulation Results - Min Attack on Actuator, u_k.

(e) *Max Attack on the Sensor, y_k*: In this scenario, we introduce a max integrity attack on plant sensor. We assume the sensor signal from the plant, y_k, is compromised and during the attack duration, the attacker replaces the true or actual signal exchanged with the value of y_{max}. Figure 74a shows that the presence of the attack results in the degraded reference tracking performance. Figure 74b depicts the total energy computation clearly indicating the presence of the attack. From the energy plots, one can observe that as expected the max integrity attack on the sensor perturbs both the local and networked energy monitors, clearly indicating that the attack is perpetuated locally. From Figure 74b, the max attack on the actuator results in a decrease in the system energy indicating the injection of excess energy and hence can be characterize as a non-passive attack. Based on the computed energy, the injected max attack on the sensor leads to the violation of passivity of the overall NCS, in addition to the observed significant degradation in performance.



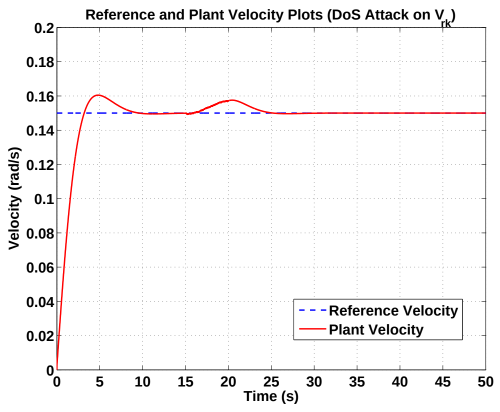
(a) Velocity Plots.



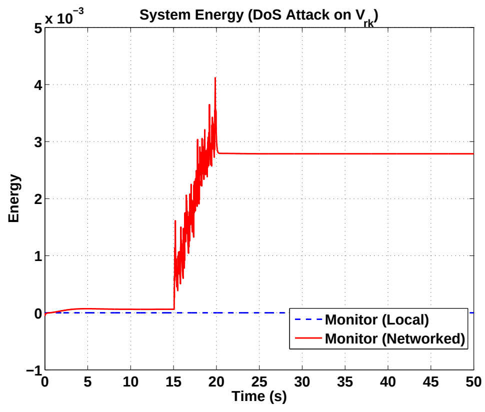
(b) Energy Balance Plot.

Figure 74: Simulation Results - Max Attack on Sensor, y_k.

3. *Denial-of-service attack on* V_{rk}: In this scenario, we introduce a denial-of-service attack on the NCS. In this case, during the attack duration, the attacker intentionally discards or erases the actual signal exchanged. During the attack duration, the attacker erases or discard the information exchanged over the network based on a simulated Bernoulli random variable, the probability of erasure for this evaluation was set at 0.2. Figure 75 shows that the presence of the attack clearly degrades the tracking performance. From Figure 75, one can observe that the denial-of-service attack can be characterized as passive attack as it leads to a positive total computed energy and this is in line with the results in Section 7.3.3 defining DoS attacks as always passive. Hence, passivity is always maintained but the performance in tracking is deteriorated.



(a) Velocity Plots.



(b) Energy Balance Plot Plots.

Figure 75: Simulation Results - DoS Attack on V_{rk}.

7.5 Experiments

In this section, we describe the experimental evaluation of the proposed energy-based attack detection. The evaluation is performed on the velocity control of a networked single robotic joint.

7.5.1 Experimental Setup

The robotic joint under consideration is powered by an AX$-$12 servo. A proportional-integral (PI) control law is used to modify the servo's behavior in order to track a desired reference velocity. Figure 76 depicts the setup for the experimental evaluation. The experiments are performed on a single computer with the nodes C1 and C2 in the figure representing the plant side (including servo interface) and controller sides of the network respectively. Matlab/Simulink's Real-Time Windows Target is used for the experiments. The servo interface, scattering transformation and local processing components run on the node C1 while the controller, the scattering transformation and the energy-based monitor run on the node C2. The communication between C1 and C2 is over the local host network using RTWT's Packet-Input and Output blocks. The plant side of the network receives the wave variable, U_{r_k} from the controller and sends the wave variable, V_{r_k} to the controller. The energy-based monitor is co-located with the controller. For comparison and also to evaluate the effect of the network, we also co-located an energy-based monitor with the plant as well.

The proposed EBAD is designed for linear systems but in this experiment the plant is a servo controlling the robotic joint. The inherent non-linearity of the servo, together with the noisy velocity estimates, makes it difficult to directly integrate the EBAD for the servo. Using system identification, we obtained an approximate linear model for the servo based on the torque input and the filtered velocity estimate from the servo. The detector's dynamics is then designed using the approximated linear model. The parameters for the linear model used for the servo are A=0.9952, B=0.0625, C=0.1214, D=0.0251. A sampling period of $T_s = 0.01$ was used for all the experiments. In the following section we presents results from the experimental evaluation.

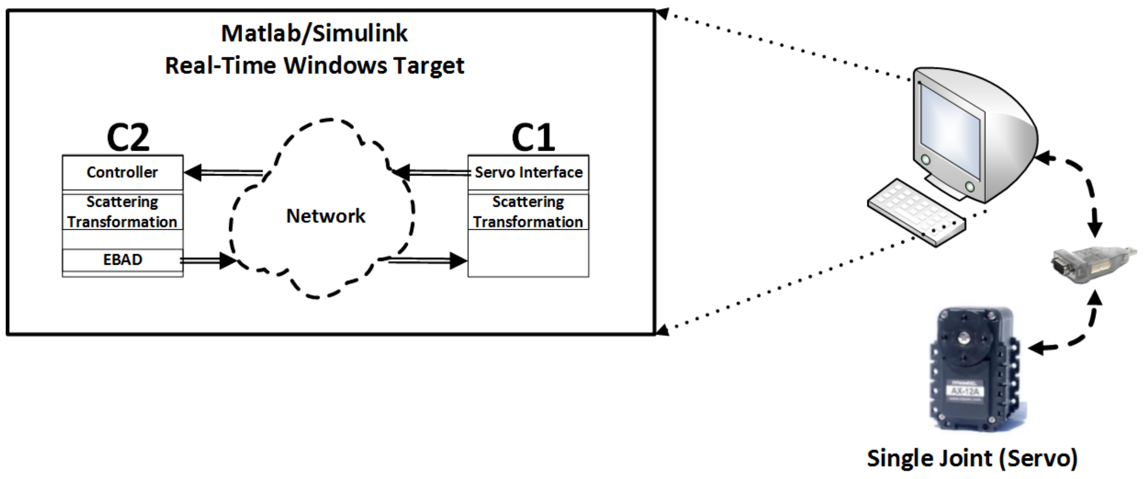


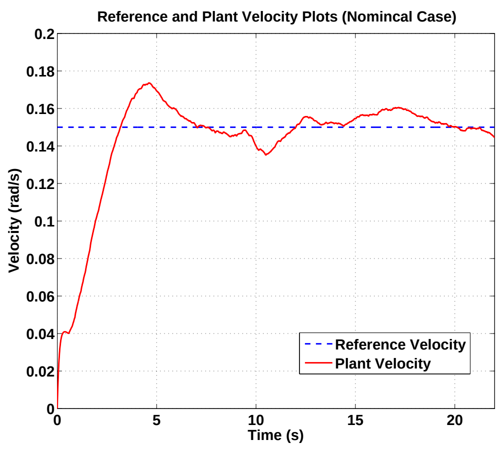
Figure 76: Experimental Setup for the Evaluation of the Energy-Based Attack Mechanism

7.5.2 Scenarios

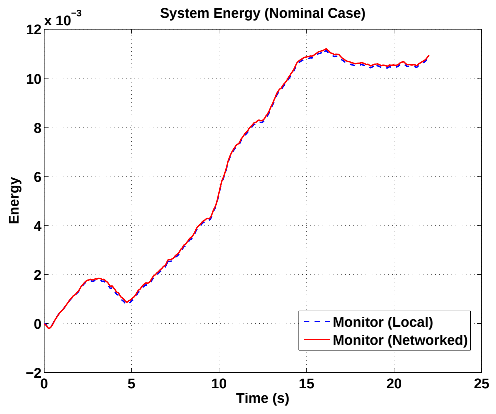
First, we present the control of the plant in the nominal case when there are no attacks. We also present the effects of the network on the system's energy. Next, we evaluate the behavior of the system under attack and the ability of the proposed approach to detect the attacks. In the experiments with attacks, the simulated attacks are injected from the duration, $t = 10s$ to $t = 15s$

1. *Nominal Case*: In this experiment, we the absence of simulated attacks on the NCS. Figure 77a depicts the reference velocity of $0.15rad/s$ as well as the plant velocity. Due to the modeling uncertainties, the controlled output tracks the reference within a bound of approximately $0.015rad/s$. Figure 77b shows the energy balance of the monitored plant computed based on the approach described in Section 7.3.2. In order to illustrate the effect of communication network on the energy-balance, we also co-located the energy-based detector at the plant side of the network to essentially perform the same total energy computation. The only difference being the delay experienced by the monitor co-located with the controller. From Figure 77b, the energy-balance computed by the local monitor and that computed by the networked monitor are almost the same with a small offset as a result of the communication network. Also, the nominal energy is not zero for the local monitor and this highlights the effect of model uncertainties of the servo that are not captured by the approximated linear model. Hence, in order the offset for the networked monitor can be used to characterize the threshold energy, E_{th}. In this characterization, when the energy level is above the bound

we can characterize the attack as being passive and when the computed energy is below this threshold we can characterize the attack as being non-passive. After multiple runs we set the E_{th} to be in the range $[-0.014, 0.014]$.



(a) Velocity Plots.



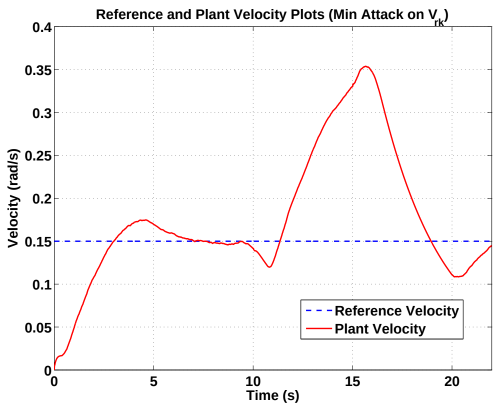
(b) Energy Balance Plot.

Figure 77: Experimental Results - Nominal Case.

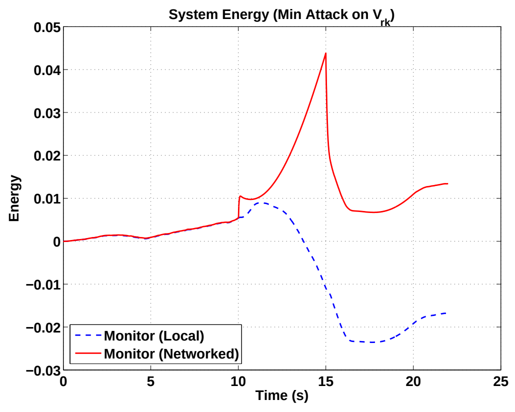
2. *Integrity Attacks*

(a) *Min Attack on V_{rk}*: In this experiment, we introduce a min integrity attack on the NCS. We assume the NCS channel from the plant to the controller, transmitting V_{rk}, is compromised and during the attack duration, the attacker replaces the true or actual signal exchanged with the value of $V_{r_{min}}$. Figure 78a shows that the presence of the attack results in the degraded reference tracking performance. Figure 78b depicts the total energy computation. The energy plot clearly indicates the presence of the attack and based on the threshold of the networked monitor such an attack is characterized as a passive attack but the local monitor shown on the same plot characterizes the attack as being non-passive. This behavior can be attributed to the uncertainties in the approximated linear model and essentially highlights the need for an accurate model for consistent attack characterization.

(b) *Max Energy Attack on V_{rk}*: In this experiment, we introduce a simulated max energy attack on the NCS. We assume the NCS channel from the plant to the controller, transmitting V_{rk}, is compromised and during the attack duration, the attacker replaces the true or actual signal exchanged with the value of V_{r_a} that maximizes the energy dissi-



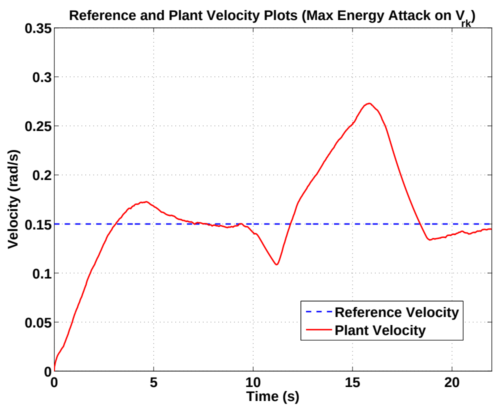
(a) Velocity Plots.



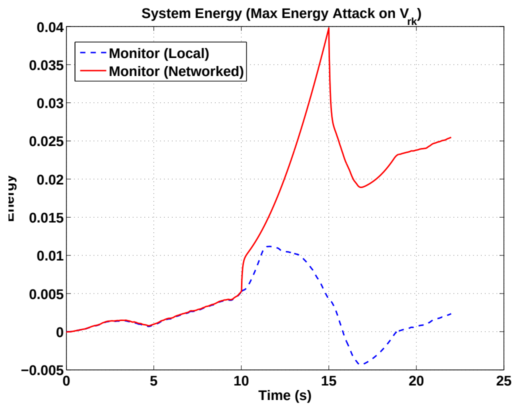
(b) Energy Balance Plot.

Figure 78: Experimental Results - Min Attack on V_{rk}.

pated at that time step. Figure 79a shows that the presence of the attack results in the degraded reference tracking performance. Figure 78b depicts the total energy computation clearly indicating the presence of the attack. Additionally, one can observe that as expected the max energy attack results in an increase in the computed total energy. Based on the computed energy, passivity of the overall NCS is still guaranteed.



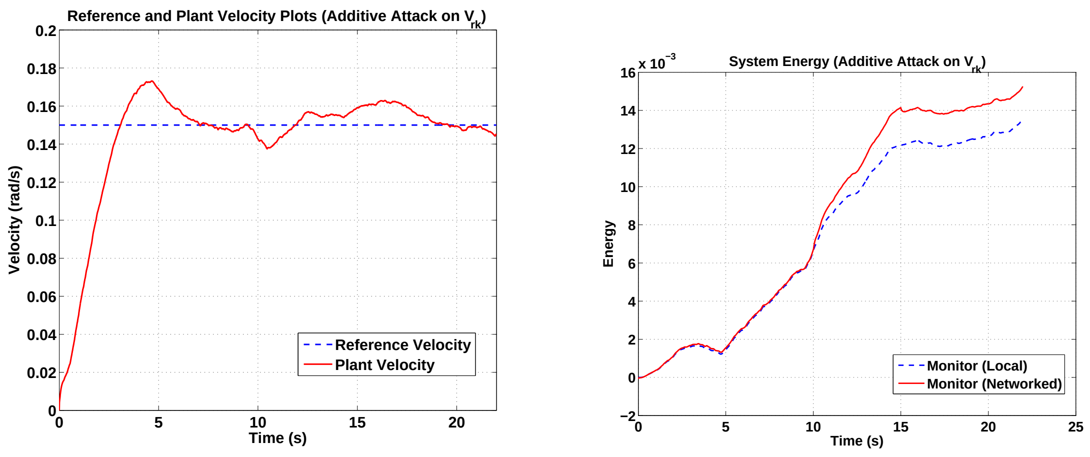
(a) Velocity Plots.



(b) Energy Balance Plot.

Figure 79: Experimental Results - Max Energy Attack on V_{rk}.

(c) *Additive Attack*: In this experiment, we introduce an additive integrity attack on the NCS. In this case, during the attack duration, the attacker introduces an additional bias, α to the true or actual signal exchanged. Figure 80a shows that the presence of the attack degrades the tracking performance slightly. Figure 80b depicts the total energy computation clearly indicating the presence of the attack. Additionally, one can observe that the additive integrity attack in this case can be characterize as a passive attack as it leads to a computed energy value larger than the nominal threshold. In this case passivity of the overall system is still guaranteed.



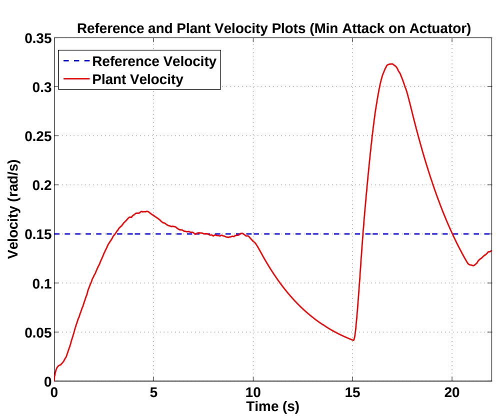
(a) Velocity Plots.

(b) Energy Balance Plot.

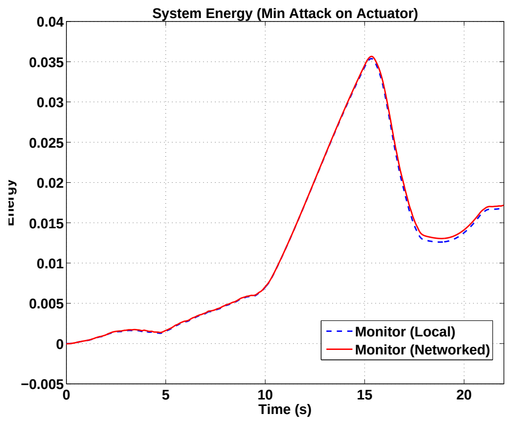
Figure 80: Experimental Results - Additive Attack on V_{rk}.

(d) *Min Attack on the Actuator,* u_k: In this experiment, we introduce a min integrity attack on actuator of the plant. We assume the actuator command into the plant, u_k, is compromised and during the attack duration, the attacker replaces the true or actual signal exchanged with the value of u_{min}. Figure 81a shows that the presence of the attack results in the degraded reference tracking performance. Figure 81b depicts the total energy computation clearly indicating the presence of the attack. From the energy plots, one can observe that as expected the min integrity attack on the actuator perturbs both the local and networked energy monitors, clearly indicating that the attack is perpetuated locally. From Figure 81b, the min attack on the actuator results in an increase in the system energy and can be characterize as a passive attack. In this case the attack does not destroy passivity of the overall NCS but significantly affects the tracking performance.

(e) *Max Attack on the Sensor,* y_k: In this scenario, we introduce a max integrity attack on plant sensor. We assume the sensor signal from the plant, y_k, is compromised and during the attack duration, the attacker replaces the true or actual signal exchanged with the value of y_{max}. Figure 82a shows that the presence of the attack results in the degraded reference tracking performance. Figure 82b depicts the total energy computation clearly indicating the presence of the attack. From the energy plots, one can observe that as ex-



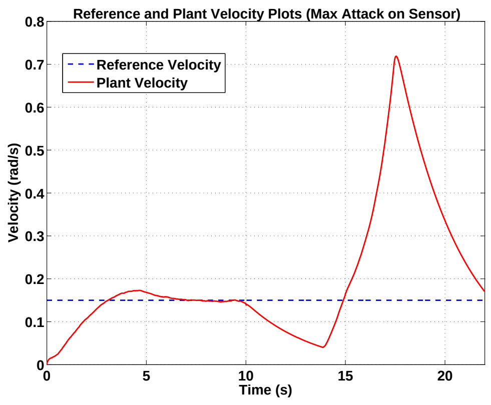
(a) Velocity Plots.



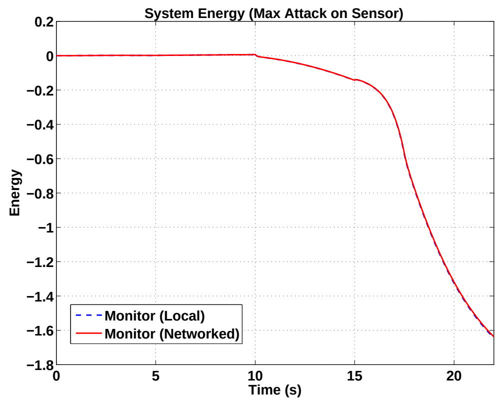
(b) Energy Balance Plot.

Figure 81: Experimental Results - Min Attack on Actuator, u_k.

pected the max integrity attack on the sensor perturbs both the local and networked energy monitors, clearly indicating that the attack is perpetuated locally. From Figure 82b, the max attack on the actuator results in a decrease in the system energy indicating the injection of excess energy and hence can be characterize as a non-passive attack. Based on the computed energy, the injected max attack on the sensor leads to the violation of passivity of the overall NCS, in addition to the observed significant degradation in performance.



(a) Velocity Plots.

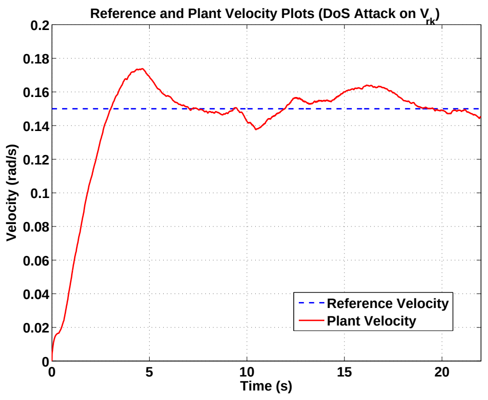


(b) Energy Balance Plot.

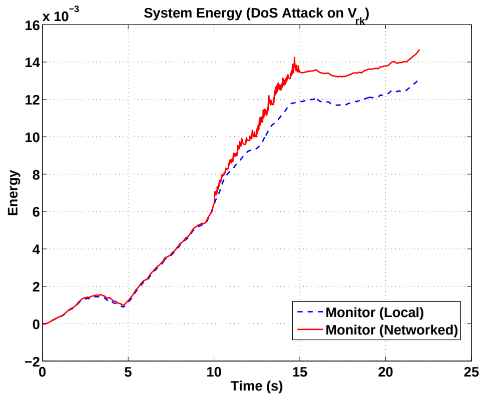
Figure 82: Experimental Results - Max Attack on Sensor, y_k.

3. *Denial-of-service attack on* V_{rk}: In this scenario, we introduce a denial-of-service attack on the NCS. In this case, during the attack duration, the attacker intentionally discards or erases the actual signal exchanged. During the attack duration, the attacker erases or discard the

information exchanged over the network based on a simulated Bernoulli random variable, the probability of erasure for this evaluation was set at 20%. Figure 83 shows that the presence of the attack. From Figure 83, one can observe that the denial-of-service attack can be characterized as passive attack as it leads to a positive total computed energy and this is in line with the results in Section 7.3.3 defining DoS attacks as always passive. Hence, passivity is always maintained but the impact of DoS is not visible in the tracking performance due to the filtering of the noisy velocity estimates.



(a) Velocity Plots.



(b) Energy Balance Plot Plots.

Figure 83: Experimental Results - DoS Attack on V_{rk}.

7.6 Summary

Due to increased attacks on CPS, there is an increased effort towards approaches to detect and secure CPS from cyber attacks. We present an energy-based attack detector for a class of CPS that are considered dissipative. We provided analytical results to show the detector can successfully detect attacks. Using well-known attack models we characterize attacks that can be considered either passive or non-passive based on their impact on the evaluated system energy. We quantitatively evaluate the performance of the proposed mechanism using simulations and experiments on a networked single joint robotic arm with the introduction of artificially simulated attacks.

CHAPTER 8

CONCLUSIONS

The increasing pervasiveness of NCS architectures in our daily lives imposes the need for their correct and dependable operation. The complexity due to the heterogeneity of these NCS architectures constrains the achievable and desirable benefits that can be derived from these systems. The work presented in this dissertation provides some fundamental results towards addressing the challenges involving the modeling, design, analysis and evaluation of dependable NCS architectures.

8.1 Summary of Contributions

The underlying principle of the work presented in this dissertation is the use of the decoupling strategies to simplify various integration challenges in construction of NCS. The contributions in this dissertation focus on addressing the fundamental challenges in NCS, including adverse network effects, complexity in modeling, analysis and evaluation of NCS, as well as ensuring the secure operation of NCS. We summarize our contributions in the following.

Compositional Modeling, Design and Analysis of NCS using Passivity. We present a DSML, PaNeCS, that simplifies the "correct-by-construction"' design of passive networked control systems. PaNeCS is integrated with a passivity component analysis tool, which together with encoded passivity constraints in the DSML structural semantics significantly reduces the burden typically involved in analyzing NCS for stability. Integrated model interpreters in PaNeCS, facilitates rapid prototyping of passivity-based NCS and allow for quick model reconfiguration, code generation and evaluation of designed NCS using simulations or performing actual experiments. Additionally, we present simulation results on the networked control of linear plants as well as experimental results on a networked multi-robot system designed using PaNeCS.

Performance-Aware Efficient Resource Utilization with Stability Guarantees. We present a passivity-based adaptive sampling framework for NCS that guarantees passivity while efficiently utilizing scarce network resources. We present a novel sample-and-hold scheme which allows variable sampling interval, while ensuring passivity. The sample-and-hold components essentially enable the decoupling of stability from the integration of performance in the design framework. The framework allows for any performance approach used for the variability of the sampling interval but for clear illustration we present the architecture for the specific case of trajectory tracking. We prove the passivity of the sample-and-hold components as well as passivity of the overall networked control architecture. Also, for the specific performance objective of tracking, we prove that the proposed architecture achieves the tracking objective while potentially using less resources based on a function of the tracking error, compared to the traditional fixed sampling approaches. Finally, we present simulation and experimental results using a case study on the trajectory tracking control of a networked robotic manipulator in a hierarchical networked control system.

Evaluation of Networked Control Systems. We present an integrated integrated modeling and simulation tool, NCSWT, for the evaluation of NCS. Simulation has been generally accepted as a powerful tool for system evaluation. NCSWT integrates Matlab/Simulink and ns-2 for modeling and simulation of NCS using the High Level Architecture (HLA) standard. We present the two parts of the tool, the design-time models and the run-time components. The design-time models use Model Integrated Computing (MIC) to define HLA-based model constructs such as federates representing the simulators and interactions representing the communication between the simulators. The design-time models represent the control system dynamics and networking system behaviors in order to facilitate the run-time simulation of a NCS. The run-time components represent the main software components and interfaces for the actual realization of a NCS simulation using the HLA framework. Our implementation of the NCSWT based on HLA guarantees accurate time synchronization and data communication. We demonstrate the capabilities of the tool using case studies and also provided an evaluation of tool.

Energy-Based Attack Detection Towards Ensuring Dependable NCS Operation. We formulate the energy-based attack detector (EBAD) for NCS. We demonstrate that EBAD solves the detection problem. Using well-known attack models we provide conditions under which attacks can be considered either passive or non-passive based on their impact on the evaluated system energy. We quantitatively evaluate the performance of the proposed mechanism using simulations. In addition, we provide experimental results to demonstrate the potential application of the detector to real-world systems.

8.2 Future directions

The work in this dissertation provides opportunities for several directions for future work. In PaNeCS, the concept of passivity indices can be integrated in the tool to improve the passivity analysis by not just indicating that a system is passive/strictly output passive/strictly input passive but also indicating the level of passivity, that is quantifying the excess or lack of passivity. This characterization can enable a domain designer in composing a suitable controller or component for integration in NCS. Also, the efficient analysis of nonlinear systems in a model-based framework in a constructive approach is an area of interest.

Application of adaptive sampling in cooperative control strategies which involves a network of agents interacting over a scarce communication network is an interesting direction to pursue. Stability guarantees and satisfaction of performance objectives is very important in these networked systems but the scarcity of networked resources is often neglected. The integration of passivity and adaptive sampling in such a framework can be exploited as a mechanism towards achieving efficient and yet robust group objectives.

The present energy-based attack monitor in the current work is formulated and modeled for linear systems. As observed from the experimental evaluation involving the servo with inherent nonlinearities and noise, a considerably accurate model of the system under control is required. A natural extension of the current approach is to consider and characterize the impact of noise and determine analytical bounds for the detector's performance. Also, extension to the case of direct application to nonlinear systems is an area of strong interest. From another perspective, the characterization of attacks as passive or non-passive using the current EBAD formulation is rather general, a more quantitatively significant approach will be to use the concept of passivity indices to characterize the condition of a system under attack based on the estimation of the system's IFP and OFP indices.

The ultimate goal of the detection approach is to provide enough information about the system's current status in order to determine if reconfiguration or mitigation needs to be applied in the presence of attacks. We would like to explore various potential mitigation strategies the can utilize the information from the detector to mitigate the impact of the attack not only for non-passive attacks but also for passive attacks which also affect overall system performance.

REFERENCES

[1] Lee, E.A.: Cyber physical systems: Design challenges. In: Proceedings of IEEE Symposium on Object Oriented Real-Time Distributed Computing. ISORC '08, Washington, DC, USA, IEEE Computer Society (2008) 363–369

[2] Antsaklis, P., Baillieul, J.: Special issue on technology of networked control systems. Proceedings of the IEEE **95**(1) (2007) 5–8

[3] Lian, F.L., Moyne, J., Tilbury, D.: Network design consideration for distributed control systems. IEEE Transactions on Control Systems Technology, **10**(2) (2002) 297–307

[4] Eyisi, E., Porter, J., Hall, J., Kottenstette, N., Koutsoukos, X., Sztipanovits, J.: Panecs: A modeling language for passivity-based design of networked control systems. In: In 2nd International Workshop on Model Based Architecting and Construction of Embedded Systems (ACES-MB 2009). (2009) 2741

[5] Eyisi, E., Porter, J., Kottenstette, N., Koutsoukos, X., Sztipanovits, J.: Panecs: A modeling language for passivity-based design of networked control systems. In: Control Automation (MED), 2011 19th Mediterranean Conference on. (2011) 1002 –1007

[6] Koutsoukos, X., Kottenstette, N., Hall, J., Eyisi, E., Leblanc, H., Porter, J., Sztipanovits, J.: A passivity approach for model-based compositional design of networked control systems. ACM Trans. Embed. Comput. Syst. **11**(4) (2013) 75:1–75:31

[7] Ledeczi, A., Bakay, A., Maroti, M., Volgyesi, P., Nordstrom, G., Sprinkle, J., Karsai, G.: Composing domain-specific design environments. Computer **34**(11) (2001) 44 –51

[8] Eyisi, E., Koutsoukos, X., Kottenstette, N.: Passivity-based trajectory tracking control with adaptive sampling over a wireless network. In: Resilient Control Systems (ISRCS), 2012 5th International Symposium on. (2012) 130–136

[9] Bai, J., Eyisi, E.P., Qiu, F., Xue, Y., Koutsoukos, X.D.: Optimal cross-layer design of sampling rate adaptation and network scheduling for wireless networked control systems. In: Cyber-Physical Systems (ICCPS), 2012 IEEE/ACM Third International Conference on. (2012) 107 –116

[10] Bai, J., Eyisi, E., Xue, Y., Koutsoukos, X.: Distributed sampling rate adaptation for networked control systems. In: Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on. (2011) 768 –773

[11] Eyisi, E., Bai, J., Riley, D., Weng, J., Yan, W., Xue, Y., Koutsoukos, X., Sztipanovits, J.: Ncswt: An integrated modeling and simulation tool for networked control systems. Simulation Modelling Practice and Theory **27**(0) (2012) 90 – 111

[12] Eyisi, E., Bai, J., Riley, D., Weng, J., Wei, Y., Xue, Y., Koutsoukos, X., Sztipanovits, J.: Ncswt: an integrated modeling and simulation tool for networked control systems. In: Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control. HSCC '12, New York, NY, USA, ACM (2012) 287–290

[13] Riley, D., Eyisi, E., Bai, J., Koutsoukos, X., Xue, Y., Sztipanovits, J.: Networked Control System Wind Tunnel (NCSWT) - An evaluation tool for networked multi-agent systems. In: Proc. of SIMUTools. (2011)

[14] LeBlanc, H., Eyisi, E., Kottenstette, N., Koutsoukos, X., Sztipanovits, J.: A passivity-based approach to deployment in multi-agent networks. In: International Conference on Informatics in Control, Automation and Robotics (ICINCO). (2010) 53–62

[15] Tipsuwan, Y., Chow, M.Y.: Control methodologies in networked control systems. Control Engineering Practice **11**(10) (2003) 1099 – 1111

[16] Huang, D., Nguang, S.K.: State feedback control of uncertain networked control systems with random time delays. IEEE Transactions on Automatic Control, **53**(3) (2008) 829–834

[17] Lazar, C., Carari, S.: A remote-control engineering laboratory. IEEE Transactions on Industrial Electronics, **55**(6) (2008) 2368–2375

[18] Tipsuwan, Y., Chow, M.Y.: Network-based controller adaptation based on qos negotiation and deterioration. In: Industrial Electronics Society, 2001. IECON '01. The 27th Annual Conference of the IEEE. Volume 3. (2001) 1794–1799

[19] Ortega, R., Spong, M.W.: Adaptive motion control of rigid robots: A tutorial. In: 27th IEEE Conf. on Decision and Control. (1988) 1575–1584

[20] Slotine, J.J.E., Li, W.: Adaptive manipulator control: A case study. IEEE Trans. on Aut. Control **33**(11) (1988) 995 –1003

[21] Tipsuwan, Y., Chow, M.Y.: Gain adaptation of networked mobile robot to compensate qos deterioration. In: IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]. Volume 4. (2002) 3146–3151

[22] Niemeyer, G., Slotine, J.J.E.: Telemanipulation with time delays. The International Journal of Robotics Research **23**(9) (2004) 873–890

[23] Hokayem, P.F., Spong, M.W.: Bilateral teleoperation: An historical survey. Automatica **42**(12) (2006) 2035 – 2057

[24] Tanner, H.G., Jadbabaie, A., Pappas, G.J.: Flocking in fixed and switching networks. IEEE Transactions on Automatic Control, **52**(5) (2007) 863 –868

[25] Jiangdagger, Z.P., Nijmeijer, H.: Tracking control of mobile robots: A case study in backstepping. Automatica **33**(7) (1997) 1393 – 1399

[26] Zampieri, S.: Trends in networked control systems. In: 17th World Congress The International Federation of Automatic Control. (2008) 2886–2894

[27] Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks **38**(4) (2002) 393 – 422

[28] Gharavi, H., Kumar, S.: Special issue on sensor networks and applications. Proceedings of the IEEE **91**(8) (2003) 1151–1153

[29] Olfati-Saber, R., Murray, R.: Consensus problems in networks of agents with switching topology and time-delays. IEEE Transactions onAutomatic Control **49**(9) (2004) 1520 – 1533

[30] Spanos, D.P., Olfati-Saber, R., Murray, R.M.: Approximate distributed kalman filtering in sensor networks with quantifiable performance. In: Proceedings of the 4th international symposium on Information processing in sensor networks. IPSN '05, Piscataway, NJ, USA, IEEE Press (2005)

[31] Olfati-saber, R.: Distributed kalman filtering and sensor fusion in sensor networks. In: Network Embedded Sensing and Control, volume LNCIS 331, Springer-Verlag (2006) 157–167

[32] Olfati-Saber, R., Fax, J., Murray, R.: Consensus and cooperation in networked multi-agent systems. Proceedings of the IEEE **95**(1) (2007) 215–233

[33] Ren, W., Beard, R.: Consensus seeking in multiagent systems under dynamically changing interaction topologies. IEEE Transactions on Automatic Control **50**(5) (2005) 655 – 661

[34] Xiao, F., Wang, L.: Consensus protocols for discrete-time multi-agent systems with time-varying delays. Automatica **44**(10) (2008) 2577 – 2582

[35] Boyd, S., Ghosh, A., Prabhakar, B., Shah, D.: Randomized gossip algorithms. IEEE Transactions on Information Theory **52**(6) (2006) 2508 – 2530

[36] Sinopoli, B., Sharp, C., Schenato, L., Schaffert, S., Sastry, S.: Distributed control applications within sensor networks. Proceedings of the IEEE **91**(8) (2003) 1235 – 1246

[37] LeBlanc, H.J., Koutsoukos, X.D.: Consensus in networked multi-agent systems with adversaries. In: Proceedings of the 14th international conference on Hybrid systems: computation and control. (HSCC '11), Chicago, IL (2011) 281–290

[38] LeBlanc, H.J., Koutsoukos, X.D.: Low complexity resilient consensus in networked multi-agent systems with adversaries. In: Proceedings of the 15th international conference on Hybrid systems: computation and control. (HSCC '12), Beijing, China (2012)

[39] Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. SIAM Journal on Computing **28** (1999) 1347–1363

[40] Klavins, E.: Communication complexity of multi-robot systems. In: Algorithmic Foundations of Robotics V. Volume 7 of Springer Tracts in Advanced Robotics. Springer Berlin / Heidelberg (2004) 275–292

[41] Martinez, S., Bullo, F., Cortes, J., Frazzoli, E.: On synchronous robotic networks;part i: Models, tasks, and complexity. Automatic Control, IEEE Transactions on **52**(12) (2007) 2199 –2213

[42] Martinez, S., Bullo, F., Cortes, J., Frazzoli, E.: On synchronous robotic networks;part ii: Time complexity of rendezvous and deployment algorithms. Automatic Control, IEEE Transactions on **52**(12) (2007) 2214 –2226

[43] Sepulchre, R., Paley, D.A., Leonard, N.E.: Stabilization of planar collective motion: All-to-all communication. Automatic Control, IEEE Transactions on **52**(5) (2007) 811 –824

[44] Jadbabaie, A., Lin, J., Morse, A.: Coordination of groups of mobile autonomous agents using nearest neighbor rules. Automatic Control, IEEE Transactions on **48**(6) (2003) 988 – 1001

[45] Olfati-Saber, R.: Flocking for multi-agent dynamic systems: algorithms and theory. IEEE Transactions on Automatic Control **51**(3) (2006) 401 – 420

[46] Gazi, V., Passino, K.: Stability analysis of swarms. Automatic Control, IEEE Transactions on **48**(4) (2003) 692 – 697

[47] Cao, Y., Fukunaga, A., Kahng, A.: Cooperative mobile robotics: Antecedents and directions. Autonomous Robots **4** (1997) 226–234

[48] Klavins, E., Ghrist, R., Lipsky, D.: A grammatical approach to self-organizing robotic systems. IEEE Transactions on Automatic Control **51**(6) (2006) 949 – 962

[49] Ganguli, A., Cortes, J., Bullo, F.: Distributed deployment of asynchronous guards in art galleries. In: American Control Conference, 2006. (2006) 6 pp.

[50] Cortes, J., Martinez, S., Bullo, F.: Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. IEEE Transactions on Automatic Control **51**(8) (2006) 1289 –1298

[51] Marshall, J., Broucke, M., Francis, B.: Formations of vehicles in cyclic pursuit. IEEE Transactions on Automatic Control **49**(11) (2004) 1963 – 1974

[52] Kelly, F.P., Maulloo, A.K., Tan, D.K.H.: Rate control for communication networks: Shadow prices, proportional fairness and stability. The Jour. of the Operational Research Society **49**(3) (1998) 237–252

[53] Low, S., Lapsley, D.: Optimization flow control. i. basic algorithm and convergence. IEEE/ACM Transactions on Networking **7**(6) (1999) 861 –874

[54] Mascolo, S.: Congestion control in high-speed communication networks using the smith principle. Automatica **35**(12) (1999) 1921 – 1935

[55] Hollot, C., Misra, V., Towsley, D., Gong, W.: Analysis and design of controllers for aqm routers supporting tcp flows. IEEE Transactions on Automatic Control **47**(6) (2002) 945 –959

[56] Deb, S., Srikant, R.: Rate-based versus queue-based models of congestion control. IEEE Transactions on Automatic Control **51**(4) (2006) 606 – 619

[57] Paganini, F., Wang, Z., Doyle, J., Low, S.: Congestion control for high performance, stability, and fairness in general networks. Networking, IEEE/ACM Transactions on **13**(1) (2005) 43 – 56

[58] Basar, T., Srikant, R.: A stackelberg network game with a large number of followers. Journal of Optimization Theory and Applications **115** (2002) 479–490

[59] Alpcan, T., Basar, T.: A globally stable adaptive congestion control scheme for internet-style networks with delay. Networking, IEEE/ACM Transactions on **13**(6) (2005) 1261 – 1274

[60] Fan, X., Arcak, M., Wen, J.T.: Robustness of network flow control against disturbances and time-delay. Systems and Control Letters **53**(1) (2004) 13 – 29

[61] Fan, X., Alpcan, T., Arcak, M., Wen, T., Baar, T.: A passivity approach to game-theoretic cdma power control. Automatica **42**(11) (2006) 1837 – 1847

[62] Yue, D., Han, Q., Lam, J.: Network-based robust control of systems with uncertainty. Automatica **41**(6) (2005) 999–1007

[63] Rangwala, S., Jindal, A., Jang, K., Psounis, K., Govindan, R.: Neighborhood-centric congestion control for multi-hop wireless mesh networks. ACM/IEEE Trans. on Networking (2011)

[64] Eryilmaz, A., Srikant, R.: Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. IEEE/ACM Trans. Netw. **15**(6) (2007) 1333–1344

[65] Yu, Y., Giannakis, G.: Cross-layer congestion and contention control for wireless ad hoc networks. IEEE Trans. Wirel. Commun. **7**(1) (2008) 37–42

[66] Xue, Y., Li, B., Nahrstedt, K.: Optimal resource allocation in wireless ad hoc networks: a price-based approach. Mobile Computing, IEEE Transactions on **5**(4) (2006) 347 – 364

[67] Gupta, R., Chow, M.Y.: Networked control system: Overview and research trends. IEEE Transactions on Industrial Electronics **57**(7) (2010) 2527–2535

[68] Schenato, L., Sinopoli, B., Franceschetti, M., Poolla, K., Sastry, S.: Foundations of control and estimation over lossy networks. Proceedings of the IEEE **95**(1) (2007) 163 –187

[69] Hespanha, J.P., Naghshtabrizi, P., Xu, Y.: A survey of recent results in networked control systems. Proceedings of the IEEE **95**(1) (2007) 138–162

[70] Havlevi, Y., Ray, A.: Integrated communication and control systems Part i-analysis. Journal of Dynamic Systems, Measurement and Control **110**(4) (1988) 367373

[71] Ray, A., Halevi, Y.: Integrated communication and control systems: Part ii-design considerations. In: Dynamic Systems, Measurement and Control. Volume 110. (1988) 374–381

[72] Liou, L., Ray, A.: Integrated communication and control systems Part iii-nonidentical sensor and controller sampling. Journal of Dynamic Systems, Measurement and Control **112**(3) (1990) 357364

[73] Luck, R., Ray, A.: An observer-based compensator for distributed delays. Automatica **26**(5) (1990) 903 – 908

[74] Nilsson, J.: Real-time control systems with delays. PhD thesis, Lund Institute of Technology (1998)

[75] Walsh, G., Beldiman, O., Bushnell, L.: Asymptotic behavior of networked control systems. In: Control Applications, 1999. Proceedings of the 1999 IEEE International Conference on. Volume 2. (1999) 1448 –1453

[76] Walsh, G., Ye, H., Bushnell, L.: Stability analysis of networked control systems. In: American Control Conference, 1999. Proceedings of the 1999. Volume 4. (1999) 2876 –2880

[77] Hong, S.: Scheduling algorithm of data sampling times in the integrated communication and control systems. Control Systems Technology, IEEE Transactions on **3**(2) (1995) 225 –230

[78] Goktas, F.: ”distributed control of systems over communication networks”. PhD. dissertation, University of Pennsylvania. (2000)

[79] Almutairi, N., Chow, M.Y., Tipsuwan, Y.: Network-based controlled dc motor with fuzzy compensation. In: Industrial Electronics Society, 2001. IECON ’01. The 27th Annual Conference of the IEEE. Volume 3. (2001) 1844 –1849 vol.3

[80] Zadeh, L.: Outline of a new approach to the analysis of complex systems and decision processes. Systems, Man and Cybernetics, IEEE Transactions on **SMC-3**(1) (1973) 28 –44

[81] Nešic, D., Liberzon, D.: A Unified Framework for Design and Analysis of Networked and Quantized Control Systems. IEEE Transactions on Automatic Control **54**(4) (2009) 732–747

[82] Brockett, R., Liberzon, D.: Quantized feedback stabilization of linear systems. IEEE Transactions on Automatic Control **45**(7) (2000) 1279–1289

[83] Skaf, J., Boyd, S.: Analysis and synthesis of state-feedback controllers with timing jitter. IEEE Transactions on Automatic Control **54**(3) (2007) 652 – 657

[84] Bhave, A., Krogh, B.: Performance bounds on state-feedback controllers with network delay. In: 47th IEEE Conference on Decision and Control. (2008) 4608–4613

[85] Pajic, M., Sundaram, S., Pappas, G., Mangharam, R.: The wireless control network: A new approach for control over networks. IEEE Transactions on Automatic Control **56**(10) (2011) 2305 –2318

[86] Cloosterman, M., Hetel, L., van de Wouw, N., Heemels, W., Daafouz, J., Nijmeijer, H.: Controller synthesis for networked control systems. Automatica **46**(10) (2010) 1584 – 1594

[87] Li, H., Sun, Z., Chow, M.Y., Sun, F.: Gain-scheduling-based state feedback integral control for networked control systems. IEEE Transactions on Industrial Electronics **58**(6) (2011) 2465 –2472

[88] Onat, A., Naskali, T., Parlakay, E., Mutluer, O.: Control over imperfect networks: Model-based predictive networked control systems. IEEE Transactions on Industrial Electronics **58**(3) (2011) 905 –913

[89] Zhao, Y.B., Liu, G.P., Rees, D.: Design of a packet-based control framework for networked control systems. IEEE Transactions on Control Systems Technology **17**(4) (2009) 859 –865

[90] Funda, J., Paul, R.P.: A Symbolic Teleoperator Interface For Time-delayed Underwater Robot Manipulation. ’Ocean Technologies and Opportunities in the Pacific for the 90’s’. (1991)

[91] Yoon, W.K., Goshozono, T., Kawabe, H., Kinami, M., Tsumaki, Y., Uchiyama, M., Oda, M., Doi, T.: Model-based space robot teleoperation of ets-vii manipulator. Robotics and Automation, IEEE Transactions on **20**(3) (2004) 602 – 612

[92] Madhani, A., Niemeyer, G., Salisbury, J.K., J.: The black falcon: a teleoperated surgical instrument for minimally invasive surgery. In: Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on. Volume 2. (1998) 936 –944

[93] Lim, J., Ko, J., Lee, J.: Internet-based teleoperation of a mobile robot with force-reflection. In: Control Applications, 2003. CCA 2003. Proceedings of 2003 IEEE Conference on. Volume 1. (2003) 680 – 685

[94] Wei, W., Kui, Y.: Teleoperated manipulator for leak detection of sealed radioactive sources. In: Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on. Volume 2. (2004) 1682 – 1687

[95] Burdea, G., Coiffet, P.: Virtual Reality Technology. John Wiley, New York (1994)

[96] Niemeyer, G., Slotine, J.J.: Stable adaptive teleoperation. IEEE Journal of Oceanic Engineering **16**(1) (1991) 152–162

[97] Berestesky, P., Chopra, N., Spong, M.W.: Discrete time passivity in bilateral teleoperation over the internet. Proceedings IEEE International Conference on Robotics and Automation (ICRA) (5) (2004) 4557–4564

[98] Chopra, N., Berestesky, P., Spong, M.: Bilateral teleoperation over unreliable communication networks. IEEE Transactions on Control Systems Technology **16**(2) (2008) 304–313

[99] van der Schaft, A.: L2-Gain and Passivity in Nonlinear Control. Springer-Verlag New York, Inc., Secaucus, NJ, USA (1999)

[100] Stramigioli, S., Secchi, C., van der Schaft, A.J., Fantuzzi, C.: Sampled data systems passivity and discrete port-Hamiltonian systems. IEEE Transactions on Robotics **21**(4) (2005) 574 – 587

[101] Hannaford, B., Ryu, J.H.: Time-domain passivity control of haptic interfaces. IEEE Transactions on Robotics and Automation **18**(1) (2002) 1–10

[102] Gao, H., Chen, T., Chai, T.: Passivity and passification of networked control systems. SIAM Journal of Control and Optimization **46**(4) (2008) 1299–1322

[103] Matiakis, T., Hirche, S., Buss, M.: The scattering transformation for networked control systems. In: Proceedings of the IEEE Conference on Control Applications (CCA). (2005) 705–710

[104] Chopra, N.: Passivity results for interconnected systems with time delay. In: 47th IEEE Conference on Decision and Control. (2008)

[105] Hirche, S., Matiakis, T., Buss, M.: A distributed controller approach for delay-independent stability of networked control systems. Automatica **45**(8) (2009) 1828–1836

[106] Zames, G.: On the input-output stability of time-varying nonlinear feedback systems. i. conditions derived using concepts of loop gain, conicity and positivity. IEEE Transactions on Automatic Control **AC-11**(2) (1966) 228–238

[107] Kottenstette, N., Koutsoukos, X., Hall, J., Sztipanovits, J., Antsaklis, P.: Passivity-based design of wireless networked control systems for robustness to time-varying delays. In: Real-Time Systems Symposium, Washington, DC, USA (2008) 15–24

[108] Kottenstette, N., LeBlanc, H., Eyisi, E., Koutsoukos, X.: Multi-rate networked control of conic (dissipative) systems. In: American Control Conference (ACC). (2011) 274–280

[109] Kottenstette, N., Antsaklis, P.: Control of multiple networked passive plants with delays and data dropouts. In: American Control Conference, Seattle, Washington (2008) 3126–3132

[110] Fettweis, A.: Wave digital filters: theory and practice. Proceedings of the IEEE **74**(2) (1986) 270 – 327

[111] Kottenstette, N., Hall, J., Koutsoukos, X., Antsaklis, P., Sztipanovits, J.: Digital control of multiple discrete passive plants over networks. International Journal of Systems, Control and Communications **3**(2) (2011) 194–228

[112] Kottenstette, N., Chopra, N.: Lm2-stable digital-control networks for multiple continuous passive plants. 1st IFAC Workshop on Estimation and Control of Networked Systems (NecSys'09) (2009) 120–125

[113] LeBlanc, H.J., Eyisi, E., Kottenstette, N., Koutsoukos, X.D., Sztipanovits, J.: Synchronized deployment of networked multi-agent systems with network uncertainties: A passivity approach. Journal of Control Science and Engineering (2011) submitted and under review.

[114] Franklin, G.F., Powel, J.D., Workman, M.L.: Digital Control of Dynamic Systems. Addison Wesley (1997)

[115] Astrom, K., Bernhardsson, B.: Comparison of riemann and lebesgue sampling for first order stochastic systems. In: 41st IEEE Conf. on Decision and Control. (2002) 2011 – 2016

[116] Arzen, K.: A simple event based pid controller. In: Proc. of 14th IFAC World Congress. (1999) 423 – 428

[117] Hristu-Varsakelis, D., Kumar, P.: Interrupt-based feedback control over a shared communication medium. In: Decision and Control, 2002, Proceedings of the 41st IEEE Conference on. Volume 3. (2002) 3223 – 3228

[118] Tabuada, P., Wang, X.: Preliminary results on state-trigered scheduling of stabilizing control tasks. In: Decision and Control, 2006 45th IEEE Conference on. (2006) 282 –287

[119] Tabuada, P.: Event-triggered real-time scheduling of stabilizing control tasks. Automatic Control, IEEE Transactions on **52**(9) (2007) 1680 –1685

[120] Wang, X., Lemmon, M.: Event-triggering in distributed networked control systems. Automatic Control, IEEE Transactions on **56**(3) (2011) 586 –601

[121] Heemels, W., Donkers, M., Teel, A.: Periodic event-triggered control based on state feedback. In: Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on. (2011) 2571 –2576

[122] Otanez, P., Moyne, J., Tilbury, D.: Using deadbands to reduce communication in networked control systems. In: American Control Conference, 2002. Proceedings of the 2002. Volume 4. (2002) 3015 – 3020

[123] Yook, J., Tilbury, D., Soparkar, N.: Trading computation for bandwidth: reducing communication in distributed control systems using state estimators. Control Systems Technology, IEEE Transactions on **10**(4) (2002) 503 –518

[124] Hirche, S., Buss, M., Hinterseer, P., Steinbach, E.: Towards deadband control in networked teleoperation systems. In: IN: PROCEEDINGS OF THE 16.TH IFAC WORLD. (2005)

[125] Heemels, W.P.M.H., Sandee, J.H., Van Den Bosch, P.P.J.: Analysis of event-driven controllers for linear systems. International Journal of Control **81**(4) (2008) 571–590

[126] Ploennigs, J., Vasyutynskyy, V., Kabitzsch, K.: Comparative study of energy-efficient sampling approaches for wireless control networks. Industrial Informatics, IEEE Transactions on **6**(3) (2010) 416 –424

[127] Henningsson, T., Johannesson, E., Cervin, A.: Sporadic event-based control of first-order linear stochastic systems. Automatica **44**(11) (2008) 2890 – 2895

[128] Rabi, M., Johansson, K., Johansson, M.: Optimal stopping for event-triggered sensing and actuation. In: Decision and Control, 2008. CDC 2008. 47th IEEE Conference on. (2008) 3607 –3612

[129] Molin, A., Hirche, S.: Structural characterization of optimal event-based controllers for linear stochastic systems. In: Decision and Control (CDC), 2010 49th IEEE Conference on. (2010) 3227 –3233

[130] Yu, H., Antsaklis, P.J.: Event-triggered output feedback control for networked control systems using passivity: Achieving stability in the presence of communication delays and signal quantization. Automatica **49**(1) (2013) 30 – 38

[131] Dimarogonas, D., Frazzoli, E., Johansson, K.: Distributed event-triggered control for multi-agent systems. Automatic Control, IEEE Transactions on **57**(5) (2012) 1291 –1297

[132] Seyboth, G., Dimarogonas, D., Johansson, K.: Control of multi-agent systems via event-based communication. In: 18th IFAC World Congress. (2011) 10086–10091

[133] Mazo, M., Tabuada, P.: Decentralized event-triggered control over wireless sensor/actuator networks. Automatic Control, IEEE Transactions on **56**(10) (2011) 2456 –2461

[134] Garcia, E., Antsaklis, P.: Adaptive stabilization of model-based networked control systems. In: American Control Conference (ACC), 2011. (2011) 1094 –1099

[135] Lehmann, D.: Event-based state-feedback control. PhD thesis, Ruhr-Universitat Bochum (2011)

[136] Velasco, M., Fuertes, J.M., Marti, P.: The self triggered task model for real-time control systems. In: 24th IEEE Real-Time Systems Symposium (work in progress). (2003) 67–70

[137] Jr., M.M., Anta, A., Tabuada, P.: An iss self-triggered implementation of linear controllers. Automatica **46**(8) (2010) 1310 – 1314

[138] Anta, A., Tabuada, P.: To sample or not to sample: Self-triggered control for nonlinear systems. Automatic Control, IEEE Transactions on **55**(9) (2010) 2030 –2042

[139] Wang, X., Lemmon, M.: Self-triggered feedback control systems with finite-gain l2 stability. Automatic Control, IEEE Transactions on **54**(3) (2009) 452 –467

[140] Brockett, W.: Minimum attention control. In: Decision and Control, 1997., Proceedings of the 36th IEEE Conference on. Volume 3. (1997) 2628 –2632

[141] Anta, A., Tabuada, P.: On the minimum attention and anytime attention problems for nonlinear systems. In: Decision and Control (CDC), 2010 49th IEEE Conference on. (2010) 3234 –3239

[142] Donkers, M., Tabuada, P., Heemels, W.: On the minimum attention control problem for linear systems: A linear programming approach. In: Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on. (2011) 4717 –4722

[143] Fontanelli, D., Greco, L., Bicchi, A.: Anytime control algorithms for embedded real-time systems. In: Proceedings of the 11th international workshop on Hybrid Systems: Computation and Control. HSCC '08 (2008) 158–171

[144] Gupta, V.: On an anytime algorithm for control. In: Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on. (2009) 6218 –6223

[145] Blind, R., Allgöwer, F.: Analysis of networked event-based control with a shared communication medium: Part i - pure aloha. In: Proc. 18th IFAC World Congress, Milan, Italy (2011) 10092–10097

[146] Blind, R., Allgöwer, F.: Analysis of networked event-based control with a shared communication medium: Part ii - slotted aloha. In: Proc. 18th IFAC World Congress, Milan, Italy (2011) 8830–8835

[147] Ramesh, C., Sandberg, H., Bao, L., Johansson, K.: On the dual effect in state-based scheduling of networked control systems. In: American Control Conference (ACC), 2011. (2011) 2216 –2221

[148] Lehmann, D., Lunze, J.: Extension and experimental evaluation of an event-based state-feedback approach. Control Engineering Practice **19**(2) (2011) 101 – 112

[149] Tiberi, U.: Analysis and Design of IEEE 802.15.4 Networked Control Systems. PhD thesis, University of LAquila (2011)

[150] Irwin, G., Colandairaj, J., Scanlon, W.G.: An overview of wireless networks in control and monitoring. In Huang, D.S., Li, K., Irwin, G., eds.: Computational Intelligence. Volume 4114 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2006) 1061–1072

[151] Gravagne, I.A., Davis, J.M., Dacunha, J.J., Marks, R.J.: Bandwidth reduction for controller area networks using adaptive sampling. In: ICRA. IEEE International Conf. on. Volume 5. (2004) 5250–5255

[152] Gabel, O., Litz, L.: Qos-adaptive control in ncs with variable delays and packet losses - a heuristic approach. In: Decision and Control, 2004. CDC. 43rd IEEE Conference on. Volume 2. (2004) 1586 – 1591

[153] Al-Hammouri, A., Branicky, M., Liberatore, V., Phillips, S.: Decentralized and dynamic bandwidth allocation in networked control systems. In: IPDPS. (2006) 8

[154] Kawka, P., Alleyne, A.: Stability and feedback control of wireless networked systems. In: American Control Conference. Proc. of. Volume 4. (2005) 2953–2959

[155] Colandairaj, J., Irwin, G., Scanlon, W.: Wireless networked control systems with qos-based sampling. IET Control Theory Applications, **1**(1) (2007) 430 –438

[156] Bao, L., Skoglund, M., Fischione, C., Johansson, K.: Rate allocation for quantized control over noisy channels. In: WiOPT. (2009) 1–9

[157] Ploplys, N., Kawka, P., Alleyne, A.: Closed-loop control over wireless networks. Control Systems, IEEE **24**(3) (2004) 58 – 71

[158] Selic, B.: The pragmatics of model-driven development. IEEE Software **20**(5) (2003) 19–25

[159] Edwards, S., Lavagno, L., Lee, E., Sangiovanni-Vincentelli, A.: Design of embedded systems: formal models, validation, and synthesis. Proceedings of the IEEE **85**(3) (1997) 366 –390

[160] Karsai, G., Sztipanovits, J., Ledeczi, A., Bapty, T.: Model-integrated development of embedded software. Proceedings of the IEEE **91**(1) (2003)

[161] et al, J.P.: Towards model-based integration of tools and techniques for embedded control system design, verification, and implementation. In: Workshops and Symposia at MoDELS 2008, Springer LNCS 5421. (2008)

[162] Porter, J., Hemingway, G., Nine, H., VanBuskirk, C., Kottenstette, N., Karsai, G., Sztipanovits, J.: The esmol language and tools for high-confidence distributed control systems design. part 1: Language, framework, and analysis. In: Tech. Report ISIS-10-109, Vanderbilt University,. (2010)

[163] AS-2 Embedded Computing Systems Committee: Architecture analysis and design language (AADL). Technical Report AS5506, Society of Automotive Engineers (2004)

[164] Hudak J. and Feiler P.: Developing AADL models for control systems: A practitioner's guide. Technical Report CMU/SEI-2007-TR-014, CMU SEI (2007)

[165] Balarin, F., Watanabe, Y., Hsieh, H., Lavagno, L., Paserone, C., Sangiovanni-Vincentelli, A.L.: Metropolis: an integrated electronic system design environment. IEEE Computer **36**(4) (2003) 45–52

[166] Al-Hammouri, A.T.: A comprehensive co-simulation platform for cyber-physical systems. Computer Communications (0) (2012) –

[167] Cervin, A., Ohlin, M., Henriksson, D.: Simulation of neworked control systems using truetime. In: Proceedings of International Workshop on Networked Control Systems: Tolerant to Faults, Nancy, France (2007)

[168] Nutaro, J., Kuruganti, P., Miller, L., Mullen, S., Shankar, M.: Integrated hybrid-simulation of electric power and communications systems. In: Power Engineering Society General Meeting, 2007. IEEE. (2007) 1 –8

[169] Branicky, M., Liberatore, V., Phillips, S.: Networked control system co-simulation for co-design. In: American Control Conference, 2003. Proceedings of the 2003. Volume 4. (2003) 3341 – 3346

[170] McCanne, S., Floyd., S.: The network simulator ns-2. `http://isi.edu/nsnam/ns/` (2004)

[171] Varga, A.: Omnet++ discrete event simulation system. `http://www.omnetpp.org` (2004)

[172] Al-Hammouri, A., Branicky, M., Liberatore, V., Phillips, S.: Decentralized and dynamic bandwidth allocation in networked control systems. In: Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International. (2006)

[173] MathWorks: Matlab, the language of technical computing. `http://www.mathworks.com` (2008)

[174] Modelica: Modelica and modelica association. `http://www.modelica.org` (2000)

[175] Lee, E.A., Hylands, C., Janneck, J., Davis II, J., Liu, J., Liu, X., Neuendorffer, S., Stewart, S.S.M., Vissers, K., Whitaker, P.: Overview of the ptolemy project. `http://ptolemy.eecs.berkeley.edu` (2001)

[176] Reutersward, P., Akesson, J., Cervin, A., Arzen, K.E.: Truetime network-a network simulation library for modelica. In: Proceedings of the International Modelica Conference. (2009)

[177] Baldwin, P., Kohli, S., Lee, E.A., Liu, X., Zhao, Y., Ee, C.C.T., Brooks, C.H., Krishnan, N.V., Neuendorffer, S., Zhong, C., Zhou, R.: Visualsense: Visual modeling for wireless and sensor network systems. Technical report, CiteSeerX - Scientific Literature Digital Library and Search Engine [http://citeseerx.ist.psu.edu/oai2] (United States) (2005)

[178] Kohtamaki, T., Pohjola, M., Brand, J., Eriksson, L.: Piccsim toolchain - design, simulation, and automatic implementation of wireless networked control systems. In: IEEE Conference on Networking, Sensing, and Control. (2009) 49–54

[179] Hatnik, U., Altmann, S.: Using modelsim, matlab/simulink and ns for simulation of distributed systems. International Conference on Parallel Computing in Electrical Engineering **0** (2004) 114–119

[180] Heimlich, O., Sailer, R., Budzisz, L.: Nmlab: A co-simulation framework for matlab and ns-2. In: 2010 Second International Conference on Advances in System Simulation (SIMUL). (2010) 152 –157

[181] Hasan, M., Yu, H., Carrington, A., Yang, T.: Co-simulation of wireless networked control systems over mobile ad-hoc network using simulink and opnet. IET Comm. **3**(8) (2009) 1297 –1310

[182] Branicky, A.A.H.M., Liberatore, V.: Co-simulation tools for networked control systems. Hybrid Systems Computation and Control, Lecture Notes in Computer Science **4981** (2008) 16–29

[183] Dahmann, J.S.: The high level architecture and beyond: technology challenges. In: Proceedings of the thirteenth workshop on Parallel and distributed simulation, Washington, DC, USA, IEEE Computer Society (1999) 64–70

[184] Kuhl, F., Dahmann, J., Weatherly, R.: Creating Computer Simulation Systems: An Introduction to the High Level Architecture. Prentice Hall PTR (1999)

[185] Neema, S., Bapty, T., Koutsoukos, X., Neema, H., Sztipanovits, J., Karsai, G.: Model based integration and experimentation of information fusion and c2 systems. In: The 12th International Conference on Information Fusion. (2009)

[186] Hemingway, G., Neema, H., Nine, H., Sztipanovits, J., Karsai, G.: Rapid synthesis of high-level architecture-based heterogeneous simulation: A model-based integration approach. In: Transactions of the Society for Modeling and Simulation International (SIMULATION). (2011)

[187] Karsai, G., Ledeczi, A., Neema, S., Sztipanovits, J.: The model-integrated computing tool-suite: Metaprogrammable tools for embedded control system design. In: IEEE Joint Conference CCA, ISIC and CACSD. (2006)

[188] Blochwitz, T., Otter, M., Arnold, M., Bausch, C., Claus, C., Elmqvist, H., Junghanns, A., Mauss, J., Monteiro, M., Neidhold, T., Neumerkel, D., Olsson, H., Peetz, J.V., Wolf, S.: The functional mockup interface for tool independent exchange of simulation models. In: Proceedings of the 8th International Modelica Conference. (2011)

[189] Laprie, J.C.: Dependable computing and fault tolerance : Concepts and terminology. In: Fault-Tolerant Computing, 1995, ' Highlights from Twenty-Five Years'., Twenty-Fifth International Symposium on. (1995) 2–10

[190] Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. Dependable and Secure Computing, IEEE Transactions on **1**(1) (2004) 11 – 33

[191] Caccavale, F., Villani, L.: Fault Diagnosis and Fault Tolerance for Mechatronic Systems. Springer (2003)

[192] Huo, Z., Fang, H., Yan, G.: Co-design for ncs robust fault-tolerant control. In: Industrial Technology, 2005. ICIT 2005. IEEE International Conference on. (2005) 119 –124

[193] Patton, R., Frank, P.: Issues of Fault Diagnosis for Dynamic Systems. Springer (2000)

[194] Zhang, Y., Jiang, J.: Bibliographical review on reconfigurable fault-tolerant control systems. Annual Reviews in Control **32**(2) (2008) 229 – 252

[195] Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M.: Diagnosis and Fault-Tolerant Control. Springer-Verlag (2006)

[196] Isermann, R.: Fault-tolerant systems a short introduction. In: Fault-Diagnosis Applications. Springer Berlin Heidelberg (2011) 285–289

[197] Seto, D., Krogh, B., Sha, L., Chutinan, A.: Dynamic control system upgrade using the simplex architecture. Control Systems, IEEE **18**(4) (1998) 72 –80

[198] Bodson, M., Lehoczky, J., Rajkumar, R., Sha, L., Stephan, J.: Analytic redundancy for software fault-tolerance in hard real-time systems. In Koob, G.M., Lau, C.G., eds.: Foundations of Dependable Computing. Volume 284 of The Kluwer International Series in Engineering and Computer Science. Springer US (1994) 183–212

[199] Ding, S.: Model-based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools. Springer (2008)

[200] Sobhani-Tehrani, E., Khorasani, K.: Fault Diagnosis of Nonlinear Systems Using a Hybrid Approach. Springer Berlin/Heidelberg (2009)

[201] Patton, R.: Fault detection and diagnosis in aerospace systems using analytical redundancy. Computing Control Engineering Journal **2**(3) (1991) 127 –136

[202] Sha, L., Rajkumar, R., Gagliardi, M.: Evolving dependable real-time systems. In: Aerospace Applications Conference, 1996. Proceedings., 1996 IEEE. Volume 1. (1996) 335 –346 vol.1

[203] Sha, L.: Dependable system upgrade. In: Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE. (1998) 440–448

[204] Sha, L.: Using simplicity to control complexity. Software, IEEE **18**(4) (2001) 20–28

[205] Bak, S., Chivukula, D., Adekunle, O., Sun, M., Caccamo, M., Sha, L.: The system-level simplex architecture for improved real-time embedded system safety. In: Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE. (2009) 99 –107

[206] Avizienis, A.: The n-version approach to fault-tolerant software. Software Engineering, IEEE Transactions on **SE-11**(12) (1985) 1491 – 1501

[207] Brilliant, S., Knight, J., Leveson, N.: Analysis of faults in an n-version software experiment. Software Engineering, IEEE Transactions on **16**(2) (1990) 238 –247

[208] Randell, B., Xu, J.: The evolution of the recovery block concept. In: IN SOFTWARE FAULT TOLERANCE, John Wiley and Sons Ltd (1994) 1–22

[209] Hwang, I., Kim, S., Kim, Y., Seah, C.: A survey of fault detection, isolation, and reconfiguration methods. Control Systems Technology, IEEE Transactions on **18**(3) (2010) 636–653

[210] Venkatasubramanian, V., Rengaswamy, R., Yin, K., Kavuri, S.N.: A review of process fault detection and diagnosis: Part i: Quantitative model-based methods. Computers and Chemical Engineering **27**(3) (2003) 293 – 311

[211] Frank, P.: Analytical and qualitative model-based fault diagnosis a survey and some new results. European Journal of Control **2**(1) (1996) 6 – 28

[212] Patton, R., Chen, J.: Observer-based fault detection and isolation: Robustness and applications. Control Engineering Practice **5**(5) (1997) 671 – 682

[213] Frank, P.M.: On-line fault detection in uncertain nonlinear systems using diagnostic observers: a survey. International journal of systems science **25**(12) (1994) 2129–2154

[214] Chow, E., Willsky, A.: Analytical redundancy and the design of robust failure detection systems. Automatic Control, IEEE Transactions on **29**(7) (1984) 603–614

[215] Gertler, J.J., Fang, X.W., Luo, Q.: Detection and diagnosis of plant failures; the orthogonal parity equation approach. In Leondes, C., ed.: Control & Dynamics Systems. Vol.37,Academic Press (1990) 157–216

[216] Gertler, J., Singer, D.: A new structural framework for parity equation-based failure detection and isolation. Automatica **26**(2) (1990) 381 – 388

[217] Gertler, J.J., Kunwer, M.M.: Optimal residual decoupling for robust fault diagnosis. International Journal of Control **61**(2) (1995) 395–421

[218] Ding, S.X.: Model-based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools. 1st edn. Springer Publishing Company, Incorporated (2008)

[219] Isermann, R.: Process fault detection based on modeling and estimation methodsa survey. Automatica **20**(4) (1984) 387 – 404

[220] Basseville, M., Nikiforov, I.V.: Detection of abrupt changes: Theory and application (1993)

[221] Fantuzzi, C., Secchi, C.: Energetic approach to parametric fault detection and isolation. In: American Control Conference, 2004. Proceedings of the 2004. Volume 6., IEEE (2004) 5034–5039

[222] Chen, W., Ding, S., Khan, A.Q., Abid, M.: Energy based fault detection for dissipative systems. In: Control and Fault-Tolerant Systems (SysTol), 2010 Conference on. (2010) 517–521

[223] Theilliol, D., Noura, H., Sauter, D., Hamelin, F.: Sensor fault diagnosis based on energy balance evaluation: Application to a metal processing. ISA transactions **45**(4) (2006) 603–610

[224] Yang, H., Cocquempot, V., Jiang, B.: Fault tolerance analysis for switched systems via global passivity. Circuits and Systems II: Express Briefs, IEEE Transactions on **55**(12) (2008) 1279–1283

[225] Cardenas, A., Amin, S., Sastry, S.: Secure control: Towards survivable cyber-physical systems. In: Distributed Computing Systems Workshops, 2008. ICDCS '08. 28th International Conference on. (2008) 495–500

[226] Cárdenas, A.A., Amin, S., Sastry, S.: Research challenges for the security of control systems. In: Proceedings of the 3rd conference on Hot topics in security. HOTSEC'08, Berkeley, CA, USA, USENIX Association (2008) 6:1–6:6

[227] Huang, Y.L., Cárdenas, A.A., Amin, S., Lin, Z.S., Tsai, H.Y., Sastry, S.: Understanding the physical and economic consequences of attacks on control systems. International Journal of Critical Infrastructure Protection **2**(3) (2009) 73–83

[228] Teixeira, A., Pérez, D., Sandberg, H., Johansson, K.H.: Attack models and scenarios for networked control systems. In: Proceedings of the 1st international conference on High Confidence Networked Systems. HiCoNS '12, New York, NY, USA, ACM (2012) 55–64

[229] Gupta, A., Langbort, C., Basar, T.: Optimal control in the presence of an intelligent jammer with limited actions. In: Decision and Control (CDC), 2010 49th IEEE Conference on. (2010) 1096–1101

[230] Amin, S., Cárdenas, A.A., Sastry, S.S.: Safe and secure networked control systems under denial-of-service attacks. In: Proceedings of the 12th International Conference on Hybrid Systems: Computation and Control. HSCC '09, Springer-Verlag (2009) 31–45

[231] Long, M., Wu, C.H., Hung, J.Y.: Denial of service attacks on network-based control systems: Impact and mitigation. IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS **1**(2) (2005) 85

[232] Mo, Y., Sinopoli, B.: False data injection attacks in control systems. In: 1st Workshop on Secure Control Systemss, Stockholm, Sweden (2010)

[233] Mo, Y., Sinopoli, B.: Secure control against replay attacks. In: Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on, IEEE (2009) 911–918

[234] Liu, Y., Ning, P., Reiter, M.K.: False data injection attacks against state estimation in electric power grids. ACM Transactions on Information and System Security (TISSEC) **14**(1) (2011) 13

[235] Esfahani, P., Vrakopoulou, M., Margellos, K., Lygeros, J., Andersson, G.: A robust policy for automatic generation control cyber attack in two area power network. In: Decision and Control (CDC), 2010 49th IEEE Conference on. (2010) 5973–5978

[236] Smith, R.: A decoupled feedback structure for covertly appropriating networked control systems. In: Proc. 18th IFAC World Congress, Milan, Italy (2011) 90–95

[237] Pasqualetti, F., Dorfler, F., Bullo, F.: Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design. In: Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on, IEEE (2011) 2195–2201

[238] Pasqualetti, F., Bicchi, A., Bullo, F.: Consensus computation in unreliable networks: A system theoretic approach. Automatic Control, IEEE Transactions on **57**(1) (2012) 90–104

[239] Sundaram, S., Hadjicostis, C.: Distributed function calculation via linear iterative strategies in the presence of malicious agents. Automatic Control, IEEE Transactions on **56**(7) (2011) 1495–1508

[240] LeBlanc, H.J., Zhang, H., Koutsoukos, X., Sundaram, S.: Resilient asymptotic consensus in robust networks. Selected Areas in Communications, IEEE Journal on **31**(4) (2013) 766–781

[241] Fawzi, H., Tabuada, P., Diggavi, S.: Security for control systems under sensor and actuator attacks. In: Decision and Control (CDC), 2012 IEEE 51st Annual Conference on. (2012) 3412–3417

[242] Gupta, R.A., Chow, M.Y.: Performance assessment and compensation for secure networked control systems. In: Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE, IEEE (2008) 2929–2934

[243] Zeng, W., Chow, M.Y.: Optimal tradeoff between performance and security in networked control systems based on coevolutionary algorithms. Industrial Electronics, IEEE Transactions on **59**(7) (2012) 3016–3025

[244] Spong, M.W., Hutchinson, S., Vidyasagar, M.: Robot Modeling and Control. John Wiley and Sons Inc., New York, USA (2006)

[245] Aguinaga-Ruiz, E., Zavala-Rio, A., Santibanez, V., Reyes, F.: Global trajectory tracking through static feedback for robot manipulators with bounded inputs. Control Systems Technology, IEEE Transactions on **17**(4) (2009) 934 –944

[246] Craig, J.J., Hsu, P., Sastry, S.S.: Adaptive control of mechanical manipulators. The International Journal of Robotics Research **6**(2) (1987) 16–28

[247] H.G. Sage, M.D.M., Ostertag, E.: Robust control of robot manipulators: A survey. International Journal of Control **72**(16) (1999) 1498–1522

[248] Abdallah, C., Dawson, D., Dorato, P., Jamshidi, M.: Survey of robust control for rigid robots. Control Systems, IEEE **11**(2) (1991) 24 –30

[249] Er, M.J., Gao, Y.: Robust adaptive control of robot manipulators using generalized fuzzy neural networks. Industrial Electronics, IEEE Transactions on **50**(3) (2003) 620 – 628

[250] Subudhi, B., Morris, A.: Soft computing methods applied to the control of a flexible robot manipulator. Applied Soft Computing **9**(1) (2009) 149 – 158

[251] Campion, G., Bastin, G., Dandrea-Novel, B.: Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. Robotics and Automation, IEEE Transactions on **12**(1) (1996) 47 –62

[252] Siciliano, Brunoand Khatib, O.: Springer Handbook of Robotics. Springer (2008)

[253] Samson, C., Ait-Abderrahim, K.: Feedback control of a nonholonomic wheeled cart in cartesian space. In: Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on. (1991) 1136 –1141

[254] d'Andrea Novel, B., Bastin, G., Campion, G.: Dynamic feedback linearization of nonholonomic wheeled mobile robots. In: Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on. (1992) 2527 –2532

[255] Jiang, Z.P., Nijmeijer, H.: A recursive technique for tracking control of nonholonomic systems in chained form. Automatic Control, IEEE Transactions on **44**(2) (1999) 265 –279

[256] Samson, C.: Control of chained systems application to path following and time-varying point-stabilization of mobile robots. Automatic Control, IEEE Transactions on **40**(1) (1995) 64 –77

[257] Aicardi, M., Casalino, G., Bicchi, A., Balestrino, A.: Closed loop steering of unicycle like vehicles via lyapunov techniques. Robotics Automation Magazine, IEEE **2**(1) (1995) 27 –35

[258] deWit, C., Sordalen, O.: Exponential stabilization of mobile robots with nonholonomic constraints. Automatic Control, IEEE Transactions on **37**(11) (1992) 1791 –1797

[259] M'Closkey, R., Murray, R.: Exponential stabilization of driftless nonlinear control systems using homogeneous feedback. Automatic Control, IEEE Transactions on **42**(5) (1997) 614 –628

[260] Gregor, K., Igor, k.: Tracking-error model-based predictive control for mobile robots in real time. Robotics and Autonomous Systems **55**(6) (2007) 460 – 469

[261] Blazic, S.: A novel trajectory-tracking control law for wheeled mobile robots. Robotics and Autonomous Systems **59**(11) (2011) 1001 – 1007

[262] Wai, R.J., Liu, C.M.: Design of dynamic petri recurrent fuzzy neural network and its application to path-tracking control of nonholonomic mobile robot. Industrial Electronics, IEEE Transactions on **56**(7) (2009) 2667 –2683

[263] Dudek, G., Jenkin, M., Milios, E., Wilkes, D.: A taxonomy for swarm robots. In: Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on. Volume 1. (1993) 441 –447

[264] Sahin, E.: Swarm robotics: From sources of inspiration to domains of application. In Sahin, E., Spears, W., eds.: Swarm Robotics. Volume 3342 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2005) 10–20

[265] Mossinger, J.: An insight into the hardware and software complexity of ecus in vehicles. Advances in Computing and Information Technology, Communications in Computer and Information Science **198** (2011) 99–106

[266] Mossinger, J.: Software in automotive systems. IEEE Software **27**(2) (2010) 92 –94

[267] Michailidis, A., Spieth, U., Ringler, T., Hedenetz, B., Kowalewski, S.: Test front loading in early stages of automotive software development based on autosar. In: Design, Automation Test in Europe Conference Exhibition (DATE). (2010) 435 –440

[268] Sangiovanni-Vincentelli, A.: Electronic-system design in the automobile industry. Micro, IEEE **23**(3) (2003) 8 – 18

[269] Cook, J., Kolmanovsky, I., McNamara, D., Nelson, E., Prasad, K.: Control, computing and communications: Technologies for the twenty-first century model t. Proceedings of the IEEE **95**(2) (2007) 334 –355

[270] Sangiovanni-Vincentelli, A., Di Natale, M.: Embedded system design for automotive applications. Computer **40**(10) (2007) 42 –51

[271] Siciliano, B., Khatib, O.: Fundamentals of Vehicle Dynamics. Society of Automotive Engineers Inc (1992)

[272] Rajamani, R.: Vehicle Dynamics and Control. Springer (2005)

[273] Broy, M., Chakraborty, S., Goswami, D., Ramesh, S., Satpathy, M., Resmerita, S., Pree, W.: Cross-layer analysis, testing and verification of automotive control software. In: Proc. of the 9th ACM Intl. Conf. on Embedded software(EMSOFT). (2011) 263–272

[274] Zander, J., Schieferdecker, I., Mosterman, P.J.: Model-Based Testing for Embedded Systems: Model-Based Testing in Embedded Automotive System. CRC Press (2011)

[275] Drolia, U., Wang, Z., Pant, Y., Mangharam, R.: Autoplug: An automotive test-bed for electronic controller unit testing and verification. In: Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on. (2011) 1187 –1192

[276] Hu, W.W., Wang, M.L., Lin, Y.H.: On the software-based development and verification of automotive control systems. In: 33rd Annual Conf. of the IEEE Industrial Electronics Society. (2007) 857 –862

[277] Ray, A., Morschhaeuser, I., Ackermann, C., Cleaveland, R., Shelton, C., Martin, C.: Validating automotive control software using instrumentation-based verification. In: 24th IEEE/ACM Intl. Conference on Automated Software Engineering. (2009) 15 –25

[278] Navet, N., Song, Y., Simonot-Lion, F., Wilwert, C.: Trends in automotive communication systems. Proceedings of the IEEE **93**(6) (2005) 1204 –1223

[279] Desoer, C., Vidyasagar, M.: Feedback Systems: Input-Output Properties. Academic Press, Inc. (1975)

[280] Khalil, H.: Nonlinear Systems. Prentice-Hall, Inc., Upper Saddle River, NJ (2002)

[281] Guilemin, E.A.: Synthesis of Passive Networks. Wiley New York, Inc. (1957)

[282] Willems, J.C.: Dissipative dynamical systems part i: General theory. Archive for Rational Mechanics and Analysis **45** (1972) 321–351

[283] Willems, J.C.: The Analysis of Feedback Systems. MIT Press, Cambridge, MA, USA (1971)

[284] Willems, J.C.: Dissipative dynamical systems part ii: Linear systems with quadratic supply rates. Archive for Rational Mechanics and Analysis **45** (1972) 352–393 10.1007/BF00276494.

[285] Marquez, H.: Nonlinear Control Systems: Analysis and Design. Wiley-InterScience, Hoboken, NJ (2003)

[286] Byrnes, C.I., Lin, W.: Losslessness, feedback equivalence, and the global stabilization of discrete-time nonlinear systems. Automatic Control, IEEE Transactions on **39**(1) (1994) 83–98

[287] Haddad, W.M., Chellaboina, V.: Nonlinear dynamical systems and control: a Lyapunov-based approach. Princeton University Press (2011)

[288] Hill, D., Moylan, P.: The stability of nonlinear dissipative systems. Automatic Control, IEEE Transactions on **21**(5) (1976) 708–711

[289] Willems, J.C.: Dissipative dynamical systems part ii: Linear systems with quadratic supply rates. Archive for Rational Mechanics and Analysis **45**(5) (1972) 352–393

[290] Goodwin, G.C., Sin, K.S.: Adaptive filtering prediction and control. Courier Dover Publications (2013)

[291] Ortega, R., Van Der Schaft, A., Mareels, I., Maschke, B.: Putting energy back in control. Control Systems, IEEE **21**(2) (2001) 18–33

[292] Byrnes, C., Lin, W.: Losslessness, feedback equivalence, and the global stabilization of discrete-time nonlinear systems. Automatic Control, IEEE Transactions on **39**(1) (1994) 83 –98

[293] Byrnes, C., Isidori, A., Willems, J.: Passivity, feedback equivalence, and the global stabilization of minimum phase nonlinear systems. IEEE Transactions on Automatic Control **36**(11) (1991/11/) 1228 – 40

[294] van der Schaft, A.: L2-Gain and Passivity in Nonlinear Control. Lecture Notes in Control and Information Sciences. Springer-Verlag, Secaucus, NJ (2000)

[295] Bao, J., Lee, P.: Process Control: The Passive Systems Approach. Advances in Industrial Control. Springer (2007)

[296] Kottenstette, N., Antsaklis, P.: Relationships between positive real, passive dissipative, amp; positive systems. In: American Control Conference (ACC), 2010. (2010) 409 –416

[297] Brogliato, B., Lozano, R., Maschke, B., Egeland, O., et al.: Dissipative systems analysis and control: theory and applications. (2007)

[298] Hitz, L., Anderson, B.D.O.: Discrete positive-real functions and their application to system stability. Electrical Engineers, Proceedings of the Institution of **116**(1) (1969) 153–155

[299] Popov, V.M.: Hyperstability of Control Systems. Springer-Verlag New York, Inc. (1973)

[300] Sepulchre, R., Jankovic, M., Kokotovic, P.: Constructive Nonlinear Control. Springer-Verlag (1997)

[301] Wen, J.T.: Robustness analysis based on passivity. In: American Control Conference. (1988) 1207–1213

[302] Bao, J., Chan, K.H., Zhang, W.Z., Lee, P.L.: An experimental pairing method for multi-loop control based on passivity. Journal of Process Control **17**(10) (2007) 787 – 798

[303] Byrnes, C., Isidori, A., Willems, J.: Passivity, feedback equivalence, and the global stabilization of minimum phase nonlinear systems. Automatic Control, IEEE Transactions on **36**(11) (1991) 1228–1240

[304] Jiang, Z.P., Hill, D.J., Fradkov, A.L.: A passification approach to adaptive nonlinear stabilization. Systems and Control Letters **28**(2) (1996) 73 – 84

[305] Jadbabaie, A., Abdallah, C.T.: (Simultaneous passification and stabilization of a class of nonlinear minimum phase systems via static output feedback)

[306] Kelkar, A., Joshi, S.: Robust passification and control of non-passive systems. In: American Control Conference, 1998. Proceedings of the 1998. Volume 5. (1998) 3133–3137

[307] Kelkar, A., Joshi, S.: Robust control of non-passive systems via passification. American Control Conference **5** (1997) 2657–2661

[308] Niemeyer, G., Slotine, J.J.E.: Telemanipulation with time delays. Int. J. of Robotics Research **23**(9) (2004) 873 – 890

[309] Kottenstette, N., Hall, J., Koutsoukos, X., Antsaklis, P., Sztipanovits, J.: Digital control of multiple discrete passive plants over networks. Intl. Journal of Systems, Control and Communications, Special Issue on Progress in Networked Control Systems (2009)

[310] Ledeczi, A., Maroti, M., Bakay, A., Karsai, G., Garrett, J., Thomason, C., Nordstrom, G., Sprinkle, J., Volgyesi, P.: The generic modeling environment. Wkshp. on Intelligent Signal Processing (2001)

[311] Craig, J.J.: Introduction to Robotics: Mechanics and Control. Addison-Wesley (1989)

[312] Grant, M., Boyd, S.: Cvx: Matlab software for disciplined convex programming. http://stanford.edu/ boyd/cvx (2009)

[313] Grant, M., Boyd, S.: Graph implementations for nonsmooth convex programs. Recent Advances in Learning and Control (a tribute to M. Vidyasagar), Springer Lecture Notes in Control and Information Sciences (2008) 95–110

[314] Crustcrawler.com: Dynamixel AX-12 Manual. (2009) http://www.crustcrawler.com/products/bioloid/docs/AX-12.pdf.

[315] SIRSLab: Haptik Library Overview. (2009) http://sirslab.dii.unisi.it/haptiklibrary/overview.htm.

[316] Astrom, K.J., Wittenmark, B.: Computer Controlled Systems: Theory and Design. Prentice-Hall (1990)

[317] Dorf, R., Farren, M., Phillips, C.: Adaptive sampling frequency for sampled-data control systems. IRE Trans. on Aut. Control, **7**(1) (1962) 38 – 47

[318] Velasco, M., Fuertes, J.M., Marti, P.: The self triggered task model for real-time control systems. In: 24th IEEE Real-Time Systems Symposium (work in progress). (2003) 67–70

[319] Mazo, M., Tabuada, P.: Input-to-state stability of self-triggered control systems. In: 48th IEEE Conf. on Decision and Control. (2009) 928 –933

[320] Slotine, J.J.E., Li, W.: Applied Nonlinear Control. Prentice-Hall, Englewood Cliffs, NJ (1991)

[321] Kottenstette, N., Antsaklis, P.: Stable digital control networks for continuous passive plants subject to delays and data dropouts. 46th IEEE Conf. on Decision and Control (2007) 4433–4440

[322] MathWorks, I.T.: Real-time windows target-simulink. The Language of Technical Computing (2013)

[323] Naranjo, J.A.H., Chavarro, A.C., De Keyser, R.: A matlab® approach for implementing control algorithms in real-time: Rtwt. (2011)

[324] Zeigler, B.P., Praehofer, H., Kim, T.G., et al.: Theory of modeling and simulation. Volume 19. John Wiley New York (1976)

[325] Cervin, A., Ohlin, M., Henriksson, D.: Simulation of networked control systems using truetime. In Proceedings 3rd International Workshop on Networked Control Systems: Tolerant to Faults (2007)

[326] Karsai, G., Sztipanovits, J., Ledeczi, A., Bapty, T.: Model-integrated development of embedded software. Proceedings of the IEEE **91**(1) (2003) 145–164

[327] Neema, S., Bapty, T., Koutsoukos, X., Neema, H., Sztipanovits, J., Karsai, G.: Model based integration and experimentation of information fusion and c2 systems. In: 12th International Conference on Information Fusion. (2009)

[328] Neema, H., Nine, H., Hemingway, G., Sztipanovits, J., Karsai, G.: Rapid synthesis of multi-model simulations for computational experiments in c2. In: Armed Forces Communications and Electronics Association - George Mason University Symposium. (2009)

[329] Zegura, E.W.: GT-ITM: Georgia Tech Internetwork Topology models (software). http://www.cc.gatech.edu/project (1996)

[330] Portico: The portio project. `http://www.porticoproject.org` (2010)

[331] Riley, D., Eyisi, E., Bai, J., Xue, Y., Koutsoukos, X., Sztipanovits, J.: Networked control system wind tunnel (ncswt)- an evaluation tool for networked multi-agent systems. In: 4th Int. ICST Conf. on Simulation Tools and Techniques (SIMUTools). (2011)

[332] Leon-Garcia, A.: Probability and Random Processes for Electrical Engineering. Addison-Wesley (1993)

[333] Falliere, N., Murchu, L., (Symantec), E.C.: W32.stuxnet dossier. (2011)

[334] Journal, T.W.S.: Electricity grid in u.s. penetrated by spies. **A1** (2009)

[335] Li, C., Raghunathan, A., Jha, N.: Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system. In: 2011 13th IEEE International Conference In e-Health Networking Applications and Services (Healthcom),. (2011) 150 –156

[336] Abrams, M., Weiss, J.: Malicious control system cyber security attack case study maroochy water services. (2008)

[337] : (nerc), n. a. e. r. c. jan-june 2009 disturbance index. (2009)